COMPUTER LANGUAGE™

VOLUME 4, NUMBER 6

JUNE 1987

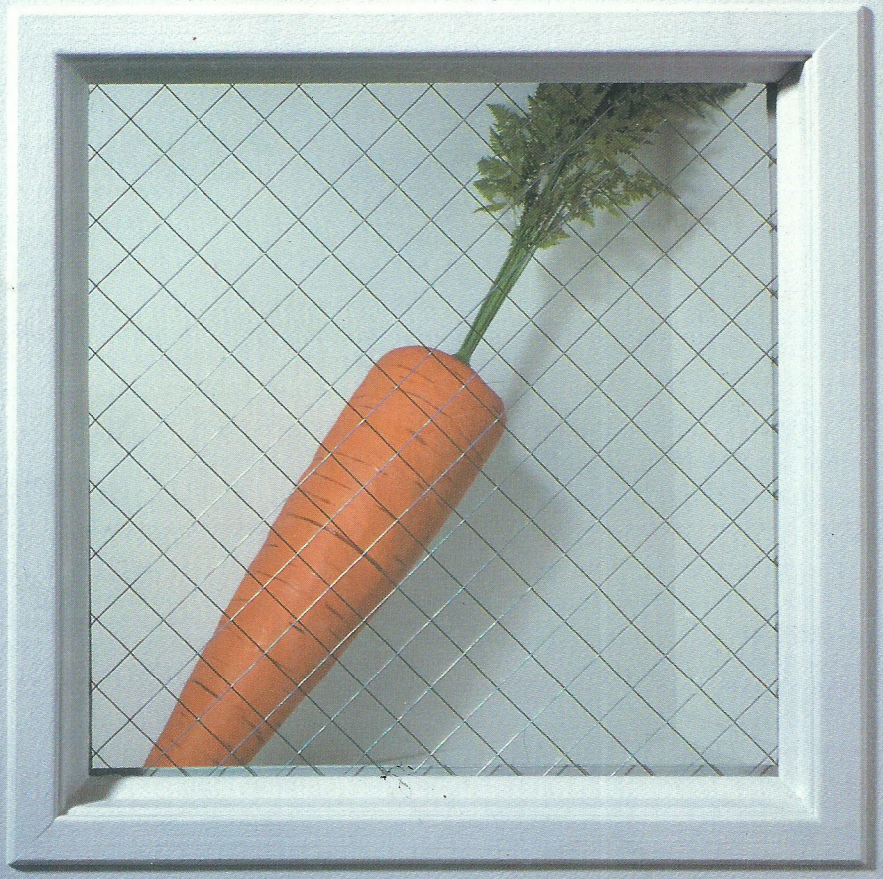
$3.50 Canada $4.50

PUSH FOR DOS MEMORY MANAGEMENT

PARSING AN EQUATION WITHOUT RECURSION

PASCAL-TO-ASSEMBLY TRANSLATOR

SOURCE-LEVEL SYMBOLIC DEBUGGER FOR UNIX



PUSH

CHARLES DUFF SHARES HIS DESIGN PHILOSOPHY

TOOLS OF THE TRADE: PROTOTYPERS AND SCREEN MA
REVIEWED

CTIVITY TOOLS

M BARRY
9244 TAVERNOR RD ZS
WILTON CA 95693

55 5383 COM034 FEB88
**********3-DIGIT 956

0 74470 17345 8

NEW

Turbo Basic®

NOW SHIPPING

Introducing Turbo Basic, the high-speed BASIC you'd expect from Borland!

It's the BASIC compiler you've been waiting for. And it's so fast that you'll never have to wait again.

Turbo Basic is a complete development environment; it includes a lightning-fast compiler, an interactive editor, and a trace debugging system.

Because Turbo Basic is compatible with BASICA, chances are that you already know how to use Turbo Basic.

With Turbo Basic your only speed is "Full Speed Ahead"!

You probably already know us for both Turbo Pascal® and Turbo Prolog.™ Well, we've done it again!

We created Turbo Basic, because BASIC doesn't have to be slow.

In fact, building fast compilers is a Borland specialty; both our Turbo Pascal and our Turbo Prolog outperform all their rivals by factors, and with Turbo Basic, we're proud to introduce the first high-speed BASIC compiler for the IBM®PC. If BASIC taught you how to walk, Turbo Basic will teach you how to run!

The Critics' Choice

"Borland has succeeded in stretching the language without weighing us down with unnecessary details . . . Turbo Basic is the answer to my wish for a simple yet blindingly fast recreational utility language . . . The one language you can't forget how to use, Turbo Basic is a computer language for the missus, the masters, the masses, and me.

Steve Gibson, InfoWorld

Borland's Turbo Basic has advantages over the Microsoft product, including support of the high-speed 8087 math chip.

John C. Dvorak"

Turbo Basic ends the basic confusion

There's now one standard: Turbo Basic.

It's fast, BASICA-compatible, and because Turbo Basic is a Borland product, the price is right, the quality is there, and the power is at your fingertips. You see, Turbo Basic's part of the fast-growing Borland family of programming languages—we call it the "Turbo Family." Hundreds of thousands of users are already using Borland's languages, so you can't go wrong. So join a whole new generation of smart IBM PC users—get your copy of Turbo Basic today. You get an easy-to-read 300+ page manual, two disks, and a free MicroCalc spreadsheet—and an instant start in the fast new world of Turbo Basic. All of this for only $99.95—Order your copy of Turbo Basic today!

Free spreadsheet included, complete with source code!

Yes, we've included MicroCalc, our sample spreadsheet, complete with source code, so that you can get started right away with a "real program." You can compile and run it "as is," or modify it.

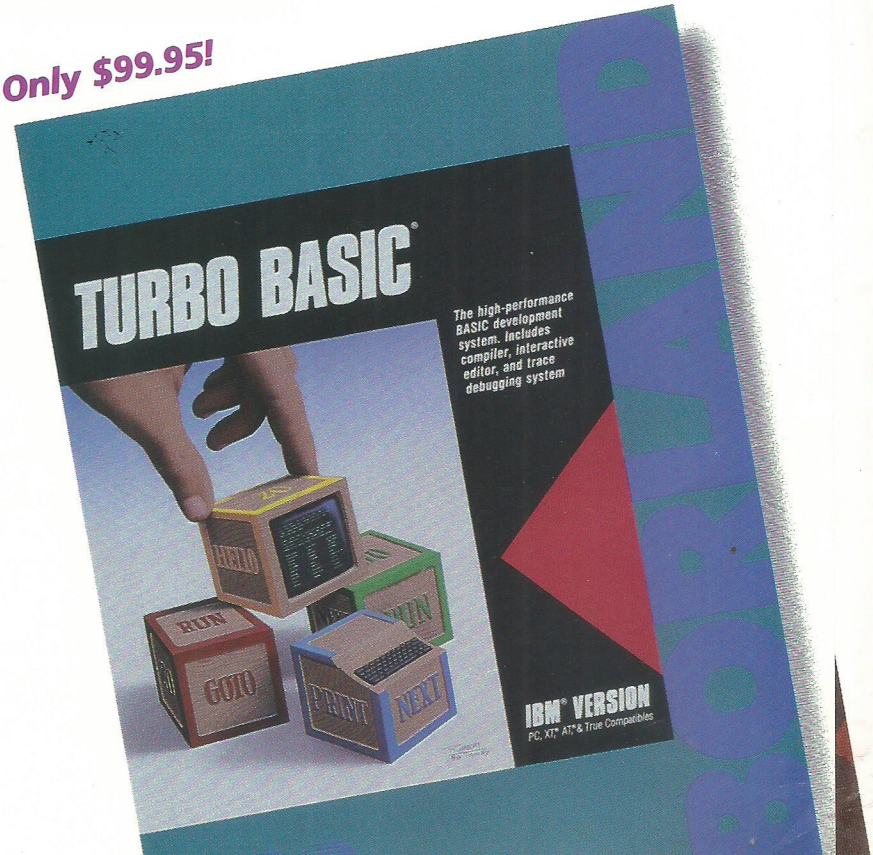
A technical look at Turbo Basic

- ☑ *Full recursion supported*
- ☑ *Standard IEEE floating-point format*
- ☑ *Floating-point support, with full 8087 (math co-processor) integration. Software emulation if no 8087 present*
- ☑ *Program size limited only by available memory (no 64K limitation)*
- ☑ *EGA and CGA support*
- ☑ *Access to local, static, and global variables*
- ☑ *Full integration of the compiler, editor, and executable program, with separate windows for editing, messages, tracing, and execution*
- ☑ *Compile, run-time, and I/O errors place you in the source code where error occurred*
- ☑ *New long integer (32-bit) data type*
- ☑ *Full 80-bit precision*
- ☑ *Pull-down menus*
- ☑ *Full window management*

System requirements

IBM PC, XT, AT and true compatibles, PC-DOS (MS-DOS) 2.0 or later. One floppy drive, 256K.

Only $99.95!

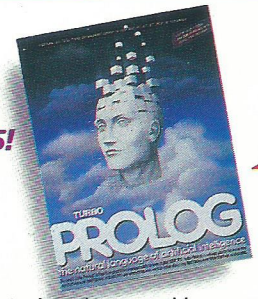


BI-1103A

Turbo Prolog™

> "If you're at all interested in artificial intelligence, databases, expert systems, or new ways of thinking about programming, by all means plunk down your $100 and buy a copy of Turbo Prolog."
> **Bruce Webster, BYTE**

Only $99.95!



Turbo Prolog, the natural language of Artificial Intelligence, is the most popular AI package in the world with more than 100,000 users. It's the 5th-generation computer programming language that brings supercomputer power to your IBM PC and compatibles. You can join the AI revolution with Turbo Prolog for only $99.95. Step-by-step tutorials, demo programs and source code included.

New! Turbo Prolog Toolbox

NEW

Our new Turbo Prolog Toolbox™ enhances Turbo Prolog—with more than 80 tools and over 8,000 lines of source code that can easily be incorporated into your programs. It includes about 40 example programs that show you how to use and incorporate your new tools.

New Turbo Prolog Toolbox features include:

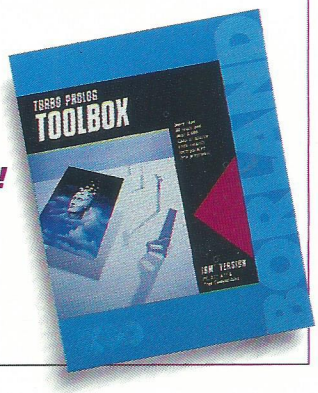
- ☑ *Business graphic generation*
- ☑ *Complete communications package*
- ☑ *File transfers from Reflex, dBASE III, 1-2-3, Symphony*
- ☑ *A unique parser generator*
- ☑ *Sophisticated user-interface design tools*

System requirements

Turbo Prolog: IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K. Turbo Prolog Toolbox requires Turbo Prolog 1.10 or higher. Dual-floppy disk drive or hard disk. 512K.

It's the complete developer's toolbox and a major addition to Turbo Prolog. You get a wide variety of menus—pull-down, pop-up, line, tree and box—so you can choose the one that suits your application best. You'll quickly and easily learn how to produce graphics; set up communications with remote devices; read information from Reflex,® dBASE III,® Lotus 1-2-3® and Symphony® files; generate parsers and design user interfaces. All of this for only $99.95.

Only $99.95!



MACINTOSH™ VERSION ALSO AVAILABLE

Turbo Pascal®

The power and high performance of Turbo Pascal is already in the hands of more than half-a-million people. The technically superior Turbo Pascal is the de facto worldwide standard and the clear leader.

The Turbo Pascal family includes:

- *Turbo Pascal® 3.0*
- *Turbo Tutor® 2.0*
- *Turbo Database Toolbox®*
- *Turbo Editor Toolbox®*
- *Turbo Graphix Toolbox®*
- *Turbo GameWorks®*
- *Turbo Pascal Numerical Methods Toolbox™*

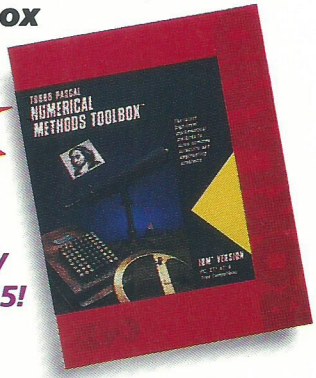
Turbo Pascal, the worldwide standard in high-speed compilers, and family.

> "The language deal of the century."
> **Jeff Duntemann, PC Magazine**

New! Turbo Pascal Numerical Methods Toolbox

NEW

Only $99.95!



What our new Numerical Methods Toolbox will do for you now:

- ☑ *Find solutions to equations*
- ☑ *Interpolations*
- ☑ *Calculus: numerical derivatives and integrals*
- ☑ *Differential equations*
- ☑ *Matrix operations: inversions, determinants and eigenvalues*
- ☑ *Least squares approximations*
- ☑ *Fourier transforms*

As well as a free demo FFT program, you also get Least Squares Fit in 5 different forms:

1. *Power*
2. *Exponential*
3. *Logarithm*
4. *5-term Fourier*
5. *5-term Polynomial*

They're all ready to compile and run.

All this for only $99.95!

System requirements

IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. Turbo Pascal 2.0 or later. Graphics module requires graphics monitor with IBM CGA, IBM EGA, or Hercules compatible adapter card, and requires Turbo Graphix Toolbox. 8087 or 80287 numeric co-processor not required, but recommended for optimal performance. 256K.

Turbo Pascal 3.0.
Includes 8087 & BCD features for 16-bit MS-DOS and CP/M-86 systems. CP/M-80 version minimum memory: 48K; 8087 and BCD features not available. 128K.

BI-1103A

NEW

Eureka: The Solver™

Anyone and everyone who routinely works with equations needs Eureka: The Solver

It solves the most complex equations in seconds. Whether you're a scientist, engineer, financial analyst, student, teacher, or some other professional, you need Eureka: The Solver!

Any problem that can be expressed as a linear or non-linear equation can be solved with Eureka. Algebra, Trigonometry and Calculus problems are a snap.

Eureka: The Solver also handles maximization and minimization problems, does plot functions, generates reports, and saves you an incredible amount of time.

X+exp(X) = 10 solved instantly instead of eventually!

Imagine you have to "solve for X," where X + exp(X) = 10, and you don't have Eureka: The Solver. What you do have is a problem, because it's going to take a lot of time guessing at "X." Maybe your guesses get closer and closer to the right answer, but it's also getting closer and closer to midnight and you're doing it the hard way.

With Eureka: The Solver, there's no guessing, no dancing in the dark—you get the right answer, right now. (PS: X = 2.0705799, and Eureka solved that one in .4 of a second!)

How to use Eureka: The Solver

It's easy.

1. *Enter your equation into the full-screen editor*
2. *Select the "Solve" command*
3. *Look at the answer*
4. *You're done*

You can then tell Eureka to

- *Evaluate your solution*
- *Plot a graph*
- *Generate a report, then send the output to your printer, disk file or screen*
- *Or all of the above*

Eureka: The Solver includes

- ☑ *A full-screen editor*
- ☑ *Pull-down menus*
- ☑ *Context-sensitive Help*
- ☑ *On-screen calculator*
- ☑ *Automatic 8087 math co-processor chip support*
- ☑ *Powerful financial functions*
- ☑ *Built-in and user-defined math and financial functions*
- ☑ *Ability to generate reports complete with plots and lists*
- ☑ *Polynomial finder*
- ☑ *Inequality solutions*

*Introductory price—good through July 1, 1987

Some of Eureka's key features

You can key in:

- ☑ *A formula or formulas*
- ☑ *A series of equations—and solve for all variables*
- ☑ *Constraints (like X has to be < or = 2)*
- ☑ *A function to plot*
- ☑ *Unit conversions*
- ☑ *Maximization and minimization problems*
- ☑ *Interest Rate/Present Value calculations*
- ☑ *Variables we call "What happens?," like "What happens if I change this variable to 21 and that variable to 27?"*

All this power for only $99.95!

Equation-solving used to be a mainframe problem, but we've solved that problem.

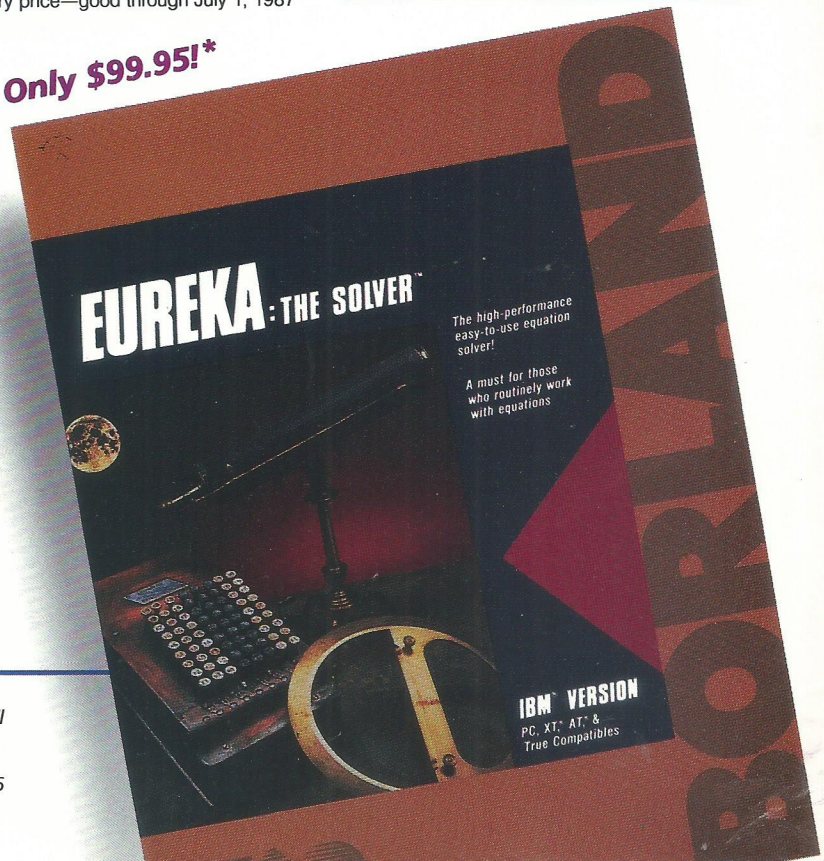
Eureka: The Solver is all you need—and it's yours for only $99.95!

That kind of savings you can calculate with your fingers!

System requirements

IBM PC, AT, XT, Portable, 3270 or true compatibles. PC-DOS (MS-DOS) 2.0 and later. 384K.

Only $99.95!*



BORLAND INTERNATIONAL

Vive la différence

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

For the dealer nearest you or to order by phone call

(800)255-8008

in CA (800) 742-1133 in Canada (800) 237-1136

BI-1103A

NEW

Turbo C®

Turbo C: The fastest, most efficient and easy-to-use C compiler at any price

Compilation speed is more than 7000 lines a minute, which makes anything less than Turbo C an exercise in slow motion. Expect what only Borland delivers: Quality, Speed, Power and Price.

Turbo C: The C compiler for amateurs and professionals

If you're just beginning and you've "kinda wanted to learn C," now's your chance to do it the easy way. Like Turbo Pascal, Turbo C's got everything to get you going.

If you're already programming in C, switching to Turbo C will considerably increase your productivity and help make your programs both smaller and faster. Actually, writing in Turbo C is a highly productive and effective method—and we speak from experience. Eureka: The Solver and our new generation of software have been developed using Turbo C.

Turbo C: a complete interactive development environment

Free MicroCalc spreadsheet with source code

Like Turbo Pascal and Turbo Prolog, Turbo C comes with an interactive editor that will show you syntax errors right in your source code. Developing, debugging, and running a Turbo C program is a snap.

Turbo C: The C compiler everybody's been waiting for. Everybody but the competition

Borland's "Quality, Speed, Power and Price" commitment isn't idle corporate chatter. The $99.95 price tag on Turbo C isn't a "typo," it's real. So if you'd like to learn C in a hurry, pick up the phone. If you're already using C, switch to Turbo C and see the difference for yourself.

System requirements

IBM PC, XT, AT and true compatibles. PC-DOS (MS-DOS) 2.0 or later. One floppy drive. 320K.

Technical Specifications

- ☑ *Compiler: One-pass compiler generating linkable object modules and inline assembler. Included is Borland's high performance "Turbo Linker." The object module is compatible with the PC-DOS linker. Supports tiny, small, compact, medium, large, and huge memory model libraries. Can mix models with near and far pointers. Includes floating point emulator (utilizes 8087/80287 if installed).*
- ☑ *Interactive Editor: The system includes a powerful, interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code.*
- ☑ *Development Environment: A powerful "Make" is included so that managing Turbo C program development is highly efficient. Also includes pull-down menus and windows.*
- ☑ *Links with relocatable object modules created using Borland's Turbo Prolog into a single program.*
- ☑ *ANSI C compatible.*
- ☑ *Start-up routine source code included.*
- ☑ *Both command line and integrated environment versions included.*

*Introductory price—good through July 1, 1987

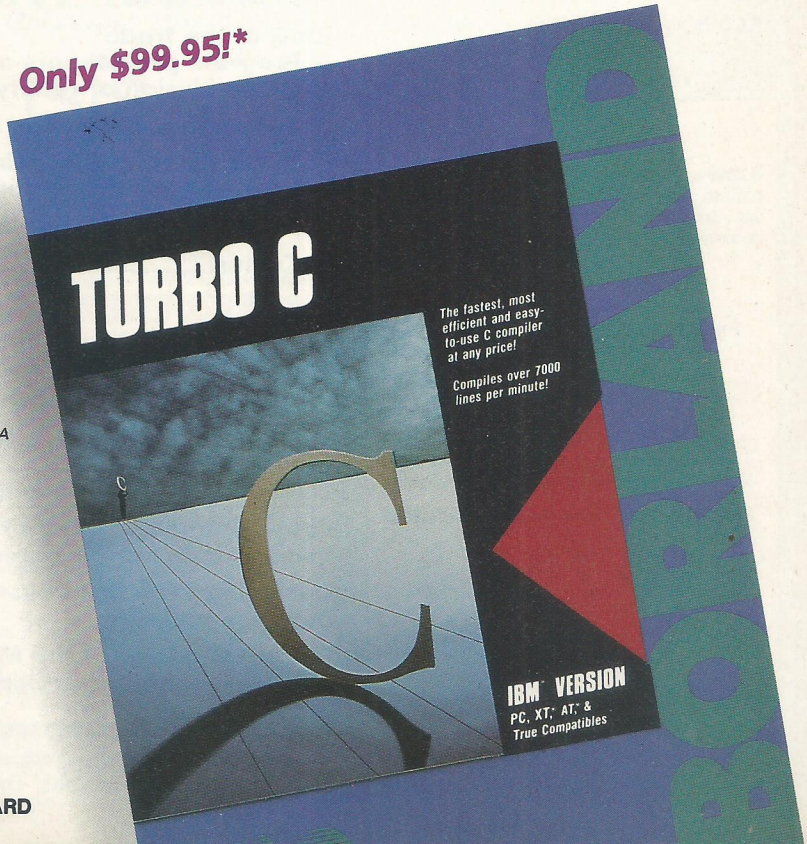
Sieve benchmark (25 iterations)

	Turbo C	Microsoft® C	Lattice C
Compile time	**3.89**	16.37	13.90
Compile and link time	**9.94**	29.06	27.79
Execution time	**5.77**	9.51	13.79
Object code size	**274**	297	301
Price	**$99.95**	$450.00	$500.00

Benchmark run on a 6 Mhz IBM AT using Turbo C version 1.0 and the Turbo Linker version 1.0; Microsoft C version 4.0 and the MS overlay linker version 3.51; Lattice C version 3.1 and the MS object linker version 3.05.

All Borland products are trademarks or registered trademarks of Borland International, Inc. or Borland/Analytica, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders. Copyright 1987 Borland International BI-1103A

Only $99.95!*



For the dealer nearest you, or to order by phone

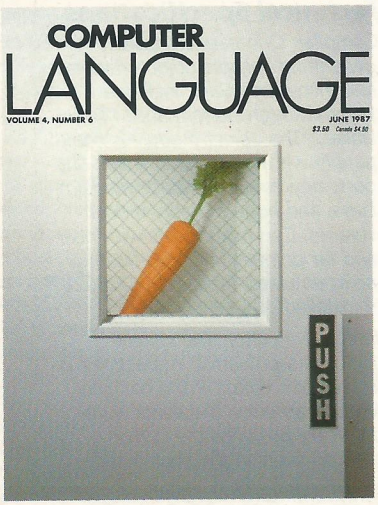
call (800) 255-8008

CA (800) 742-1133

Canada (800) 237-1136

CIRCLE 1 ON READER SERVICE CARD

COMPUTER LANGUAGE™



ARTICLES

Save Time With PUSH ____ 34
by Christopher Vergoz-Rekis
This DOS program, much like the DEC System-20 command PUSH, helps utilize operating system memory space more efficiently by pushing you to free areas without losing the process or program you're currently working on. The accompanying code shows how DOS system calls are used to gain greater control over memory, interrupts, and processes.

Parsing an Equation Without Recursion ____ 45
by Scott Magruder
The parser described in this article bypasses recursion by allowing previously defined variables and advanced functions to be used in an equation. This stand-alone version, written in Turbo Pascal, can be easily included in many other application programs.

A Source-Level Symbolic Debugger for UNIX ____ 53
by David Livshin
ddb is a debugging tool that works much like an interpreter but has the additional capabilities of on-line execution of C code, complete flow control, code patching, and full execution speed.

Reliable FORTRAN with MAT ____ 59
by Gerald Berns
A static analyzer like MAT takes the concept of the compiler one step further by first analyzing each module and then the program as a whole. Here the result is more reliable FORTRAN code.

A Translator for Lazy Programmers ____ 65
by Todd Jonathan Egan
LASM translates Pascal statements into 8086/8088 assembly language but can be adapted to meet the needs of almost any language-to-language translation project.

PRODUCT WRAP-UP

Tools of the trade
Program prototypers and the illusion of reality ____ 109
Screen managers: The sharper image ____ 115
Turbo Pascal prototyping and screen management ____ 127
New tools: COBOL screen management ____ 139
Curses! ____ 144

DEPARTMENTS

Programming on Purpose ____ 19
Divorcing data structures

ComputerVisions ____ 29
Charles Duff, ACTOR's Shakespeare

Public Domain Software Review ____ 79
Hercules cards, fast disk copies, and even more on trojans

Designing With Data Bases ____ 87
Data base integrity

Exotic Language of the Month Club ____ 97
NPL: The nonprogrammer's data base language

Product BINGO ____ 107

Software Review ____ 149
Programming Technologies' The Productivity Tree

If you thought all file managers were alike, think again. Think Btrieve.

Btrieve stands out as the one file management system for programmers who are serious about productivity. With Btrieve, your programs can use simple subroutine calls to store, retrieve, and update records. It's the file handling solution rich in functionality to meet your needs now and in the future, as your requirements grow and change.

Bfast. With Btrieve you develop fast applications, with ease. Written in assembly language for IBM PC's, Btrieve uses b-tree algorithms with automatic balancing for fast, efficient file indexing.

Bsafe. Btrieve is the only file manager with automatic file recovery. Two levels of database integrity provide complete fault tolerance in the event of accidents or power failures.

Bflexible. Develop your PC applications with the capabilities you need most. Capabilities such as: unlimited open files, unlimited records per file, 24 indexes per file and a maximum file size of 4 gigabytes. All this with Btrieve plus access from any programming language.

Bconnected. When your applications require networking, think Btrieve. A single version runs on all DOS 3 LAN's, including IBM PC Network and Novell Advanced NetWare. Btrieve is also available for Xenix and other multi-tasking operating systems.

Broyalty Free. No royalty payments on your applications. Ever.

Bsure. Btrieve comes with comprehensive documentation and unrivaled technical support for trouble-free software development. And you can add Xtrieve to supply your end users with even more, a menu-driven report writer and query system.

Think Btrieve. Make your programming performance stand out. Contact SoftCraft.

SoftCraft
A NOVELL COMPANY
P.O. Box 9802, #917
Austin, Texas 78766
(512) 346-8380 Telex 358 200

Btrieve, $245; multi-user Btrieve, $595; Xtrieve, $245; multi-user Xtrieve, $595 (for report generation, add $145 for single-user and $345 for multi-user). Requires PC-DOS or MS-DOS 2.X, 3.X or Xenix. Btrieve and Xtrieve are registered trademarks of SoftCraft Inc.

CIRCLE 2 ON READER SERVICE CARD

COMPUTER LANGUAGE

EDITOR
Regina Starr Ridley
MANAGING EDITOR
Kathy Kincade
PRODUCT REVIEW EDITOR
Bill Burch
EDITORIAL ASSISTANTS
Judy Barrows, Joanna M. Poole
TECHNICAL EDITOR
John Halamka
CONSULTING EDITOR
Craig LaGrow
CONTRIBUTING EDITORS
Judy Getts, Tim Parker,
P.J. Plauger, Matthew Rapaport
Jeffrey Star, Thomas Truitt
NATIONAL ADVERTISING DIRECTOR
Ted Bahr
(415-397-1881)
REGIONAL SALES MANAGERS
Mary Christine Gordon
(Midwest—216-493-8733)
Walter Andrzejewski
(East—617-232-5470)
Jeff Eilertsen
(West—415-397-1881)
ADVERTISING SALES ASSISTANTS
Susan McDonald, Linda Pizzotti
DIRECTOR OF PRODUCTION
Andrew A. Mickus
OPERATIONS CONSULTANT
Beatrice C. Blatteis
DIRECTOR OF CIRCULATION
Wini D. Ragus
CIRCULATION MANAGER
Donna F. Wilson
CIRCULATION MARKETING MANAGER
Jerry M. Okabe
ADVERTISING TRAFFIC COORDINATOR
Kelly B. Keller
WHOLESALE COORDINATOR
Nicola Sullivan
ART PRODUCTION
Anabella Gonzalez

PUBLISHER
Donald Pazour

CHAIRMAN OF THE BOARD
David R. Stevens
PRESIDENT
Marshall W. Freeman
VICE PRESIDENTS
Charles C. Baake
Leigh M. Freeman
Leonard E. Haas
Barbara M. Hampson
Thomas L. Kemp
Wini D. Ragus
SECRETARY/TREASURER
Miller Freeman III
ASSISTANT TREASURER
Douglas M. Denny
CORPORATE FINANCIAL MANAGER
Charles H. Benz

FOUNDING PUBLISHER
Carl Landau

MILLER FREEMAN publications

Editor's Notes

The Language Wars are here. The second half of 1987 may long be remembered as the time the major battles were fought for predominance among vendors of BASIC and C, two of the most popular and competitively priced programming languages.

The BASIC battle has begun in earnest. As of last week, the end of April, we have had production copies of Borland's TurboBASIC and Microsoft's QuickBASIC 3.0. They are currently being reviewed, compared, contrasted, and turned upside down, and you will see the results in our magazine very soon.

We'll be doing the same for the new C compilers that come out this year; for example, with Borland's Turbo C. We've already seen the market heating up with several new version announcements, and we suspect with Borland in the fray we'll see many more. Microsoft, for one, surely won't just sit by and watch. Will we see a low-end Microsoft C in the coming months too?

But the battleground isn't really on the pages of *COMPUTER LANGUAGE*. We see our role as being unbiased observers, passing on information to you for your judgments. The real battle will be fought on your computer and at your workplace.

This is going to be a very exciting six months.

The theme for this June's *COMPUTER LANGUAGE* is productivity tools. Of all the special topics we cover, this one is the most central to the goal of *COMPUTER LANGUAGE*: providing you with information and tools that will help you be more efficient and productive at work.

Here's a brief rundown on the tools presented in this month's articles:

- PUSH—A program that lets you use the DOS system calls for memory management and process control for loading the DOS command system upon demand, without interfering with the current program.
- A mathematical equation parser that doesn't use recursion.
- ddb—An interactive, source-level symbolic debugger for UNIX.
- MAT—A static analyzer (a tool that detects errors and problems in programs by analyzing their source code) for FORTRAN.
- LASM—A development environment called Lazy Assembly, for converting Pascal statements into 8086/8088 assembly language code within an editor.

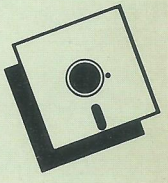
This month's product wrap-up on prototypers and screen managers should also be extremely useful to you. We take a look at a total of 22 products: four C prototypers, nine C screen managers, four Turbo Pascal prototypers/screen managers, three COBOL screen managers, and two curses packages.

Onward...

Regina Starr Ridley

Regina Starr Ridley
Editor

COMPUTER LANGUAGE code can be obtained from CompuServe (type "GO CLM" at the prompt and you'll be in our SIG), from USENET ("mod.sources"), and from several Bulletin Board Services in the U.S. and Canada. Bulletin board parameters are 8 data bits, no parity, 1 stop bit, full duplex, and 300 or 1200 baud.



(309) 342-5302	Chicago, Ill.	(403) 454-6093	Edmonton, Canada
(517) 355-3276	East Lansing, Mich.	(613) 830-1590	Ontario, Canada
(216) 486-0348	Cleveland, Ohio	(212) 980-0770	New York, N.Y.
(614) 761-1179	Columbus, Ohio	(405) 728-2463	Oklahoma City, Okla.
(206) 848-9232	Seattle, Wash.	(213) 498-6425	Long Beach, Calif.
(415) 487-0310	San Francisco, Calif.		

COMPUTER LANGUAGE *(ISSN 0749-2839) is published monthly by Miller Freeman Publications, 500 Howard St., San Francisco, CA 94105, (415) 397-1881. Please direct advertising and editorial inquiries to this address. Subscription rates for the U.S. are $29.95 for 12 issues, $39.95 for 24 issues, and $49.95 for 36 issues. Canadian/Mexican orders must be accompanied by payment in U.S. funds with additional postage of $6 per year. All other foreign must be prepaid in U.S. funds with additional postage at $15 per year for surface mail and $40 per year for air mail. All subscription orders, inquiries, and address changes should be sent to P.O. Box 11333, Des Moines, IA 50340-1333. For quickest service, telephone toll-free 1-800-341-SERV (in Iowa 1-800-233-IOWA). Second class postage is paid at San Francisco, CA and at additional mailing offices. Please allow six weeks for change of address to take effect.* COMPUTER LANGUAGE *is a registered trademark owned by the magazine's parent company, Miller Freeman Publications. All material published in* COMPUTER LANGUAGE *is*
copyrighted © 1987 by Miller Freeman Publications Inc. All rights reserved. Reproduction of material appearing in COMPUTER LANGUAGE *is forbidden without written permission.*

NEW! FROM BLAISE COMPUTING

C Tools Plus™

For The Programmer Whose Alphabet Begins & Ends With "C"

Today's programmers need more than yesterday's tools. Requirements such as removable windows and "sidekickable" pop-up utilities are changing the face of program design. You need to filter interrupts so that other resident programs still work. You need the ability to switch between multiple display pages and monitors. Today's technical demands are almost endless, but C TOOLS PLUS gives you what you need.

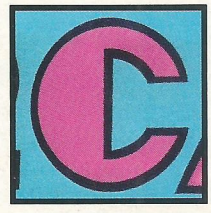
SOLID LIBRARY SUPPORT

Blaise Computing offers you solid library support that can meet all your demands and more. C TOOLS PLUS embodies the full spectrum of general-purpose utility functions that are critical to today's applications.

Here's just part of the PLUS in C TOOLS PLUS:

- **C TOOLS and C TOOLS 2** compatibility—two packages that receive rave reviews for quality, organization, usability and documentation.
- **FULL SOURCE CODE**
- **WINDOWS** that are stackable, removable, that support word wrap and that can accept user input.
- **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite resident applications.
- **MULTIPLE** monitor and display support, including EGA 43-line mode.
- **FAST DIRECT VIDEO ACCESS** for efficiency that will not constrain good program design.
- **DOCUMENTATION, TECHNICAL SUPPORT** and attention to detail that have distinguished Blaise Computing products over the years.

C TOOLS PLUS supports the Microsoft (and IBM) 3.00 and Lattice 3.00 C compilers and is just ***$175.00.***



Also Available Are:

C VIEW MANAGER *— A kit for building data entry screens and menus. Begin by designing on-screen what the operator will see; call upon our library functions from your program to display the screens and retrieve the data. Just $275, including all library source code.*

C ASYNCH MANAGER *— provides the crucial core of hardware interrupt support needed to build applications that communicate. It also includes the "XMODEM" file-transfer protocol and support for Hayes-compatible modems. All source code is included for $175.*

C TOOLS & C TOOLS 2 *— an indispensable combination still available at a low price of $175, including all source code. See review in* PC Tech Journal, *6/85.*

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

ORDER TOLL-FREE 800-227-8087!

YES, send me the PLUS I need! Enclosed is $_____ for C TOOLS PLUS. (CA residents add 6½% Sales Tax. All domestic orders add $10.00 for Federal Express shipping.)

Name: _____ Phone: (____) _____

Shipping Address: _____

City: _____ State: _____ Zip: _____

VISA or MC #: _____ Exp. Date: _____

CIRCLE 3 ON READER SERVICE CARD

The fastest, tightest code.

(Though the same can hardly be said of the name.)

We have to tell you, we had a hard time getting the name down this short.

Because Microsoft's new FORTRAN Compiler actually has a far longer list of features.

It uses the same optimizer and code generator technology that made our C Compiler the industry leader.

And we've also added special loop optimizations that give you the smallest, fastest FORTRAN code a PC can handle.

> *"Now Microsoft's FORTRAN Optimizing Compiler generates such fast code that an IBM PC/XT approaches the speed of the VAX."*
>
> *Peter Osgood, MIT, Project Athena, Director of the Real Time Lab Project.*

This compiler has already passed the toughest test there is. It's been

Microsoft FORTRAN Optimizing Compiler Version 4.0.

- Uses the Microsoft C optimizing technology, plus loop optimization to generate the fastest executable code for MS-DOS. NEW!

Execution Speed (in Seconds)	Microsoft FORTRAN v. 4.0	Ryan-McFarland FORTRAN v. 2.11	IBM Professional FORTRAN v. 1.22
Sieve	7.97	9.33	38.51
Whetstone	53.82	58.67	79.04
Lookup	5.82	18.61	26.02

- Fully GSA certified for ANSI 77 compatibility with no errors at the highest level. NEW!
- Numerous IBM VS and DEC VAX extensions. NEW!
- Microsoft CodeView: Window-oriented source-level debugger. NEW!
  - —Debug using your original source code, the resulting disassembly or both intermingled.
  - —Watch and change the values of your local and COMMON variables as you debug.
  - —Set conditional breakpoints on variables, expressions or memory; trace and single step.
  - —Debug Microsoft C programs as well as Microsoft Fortran programs.
  - —Watch and change registers and flags as you execute.
  - —Easily debug graphics oriented programs since program output is kept separate from debugger output.

GSA-certified as Full ANSI FORTRAN 77, and 100% error-free.

"The Microsoft FORTRAN Optimizing Compiler let us port the 200,000 line Boeing Mathematical Library (BCSLIB) with virtually no changes. This ANSI FORTRAN 77 code was ported directly from Cray, CDC, DEC, IBM and other mainframes and workstations."

Ivor Philips, Boeing Computer Services, Program Manager Mathematical Software Libraries.

Compiler V. 4.0 with CodeView™

We've also included the same advanced intrinsic math functions found on VAX® and IBM® VS systems. Add improvements like our new HUGE memory model, and porting the biggest mainframe programs has never been easier.

Among the many additions we've made to our package is our exclusive CodeView™ windowing debugger.

It lets you trace through programs at any level you want, from source code to assembly language.

You can open windows and watch both variables (local and COMMON) and CPU registers change.

You can set conditional breakpoints using variables and expressions.

Debugging gets even easier with the compiler's advanced diagnostics. Detailed error messages are thoroughly explained and cross-referenced in our new manuals.

Documentation that has been completely revised and expanded with tons of examples.

If we're talking your language, use one of the numbers below for more details about the Microsoft® ANSI FORTRAN 77 Optimizing Compiler Version 4.0 with CodeView, and the name of your nearest dealer.

(Even if the call's toll-free, it may be a good idea to refer to it as "FORTRAN 4.0" for short.)

Microsoft® FORTRAN

The High Performance Software.

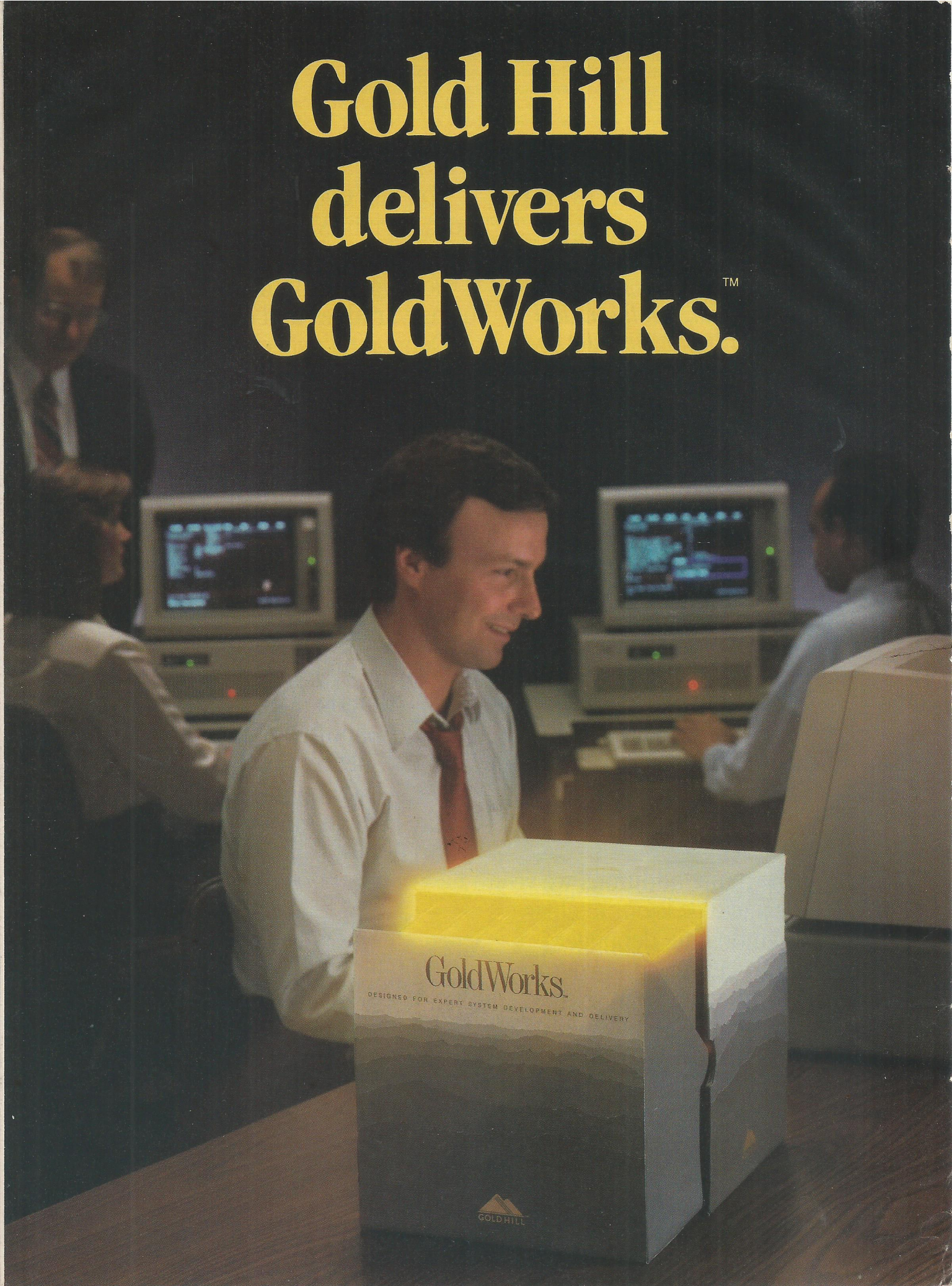
Call (800) 426-9400. In Washington State or Alaska, (206) 882-8088. In Canada, (416) 673-7638.

Microsoft and MS-DOS are registered trademarks and CodeView is a trademark of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation. VAX is a registered trademark of Digital Equipment Corporation.

- Medium, Large and Huge Memory Model Libraries. NEW!
- Mix models with NEAR, FAR and new HUGE pointers.
- Common blocks and arrays greater than 64K.
- Choose from three math libraries and generate in-line 8087/80287 instructions or floating point calls:
  - —floating point emulator (utilizes 8087/80287 if installed)
  - —8087/80287 coprocessor support
  - —alternate math package—extra speed without an 8087/80287
- Link your FORTRAN routines with Microsoft C (v. 4.0 or higher), Microsoft Pascal (v. 3.3 or higher) or Microsoft Macro Assembler.
- Largest number of 3rd party support libraries available.
- Provides more detailed diagnostic error messages (almost twice as many as competitors) and extensive documentation with non-ANSI 77 features highlighted. NEW!
- Proven reliability—tested with over 2.5 million lines of code compiled and executed.
- MS-DOS® network support with file/record locking and sharing.
- Microsoft Program Maintenance Utility rebuilds your applications after your source files have changed. NEW!
- Other utilities including faster overlay linker (links over 1Mbyte object code), library manager, EXE file compression utility, EXE file header utility, MS-DOS environment setting utility and setup utility.

CIRCLE 4 ON READER SERVICE CARD

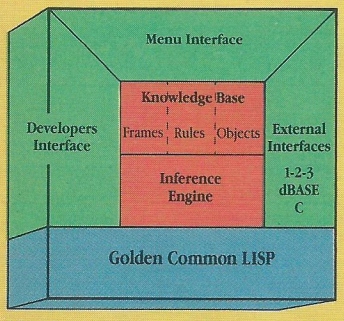
Gold Hill delivers GoldWorks.™



Now you can build and deliver serious expert systems on advanced PCs.

GoldWorks changes the economics of expert system building.

GoldWorks, formerly code-named Acorn, is designed for professional software developers who need to build serious expert systems and integrate them with conventional applications running on 286- and 386-based PCs. It combines the best features of high-end expert system tools with the ability to develop and deliver expert systems *inexpensively,* on advanced PCs.



GoldWorks is the most comprehensive expert system tool available for advanced PCs.

Works like a shell.

GoldWorks gives you the best features of an expert system shell. With the easy-to-use menu interface, you can rapidly prototype and build expert system applications without knowing the underlying programming environment. And you get the GoldWorks tutorial, the San Marco LISP Explorer® tutorial, an on-line help system, and example applications to get you started quickly.

Works like a toolkit.

GoldWorks gives you the best features of an expert system toolkit. You can access the underlying programming environment to extend and customize the system for your specific applications. And you can address up to 15 MB of extended memory on the PC AT (and even more on 386-based PCs).

Works like expert system tools previously available only on high-end workstations . . . at a fraction of the cost.

GoldWorks sets a new standard for expert system development and delivery on advanced PCs. You get frames with multiple inheritance for flexible knowledge representation. Rules supporting integrated forward and backward chaining for powerful inferencing. Object programming for developing modular applications. Plus advanced features for controlling the inferencing process, including rule sets, sponsors, rule priorities, certainty factors, and extensive rule inspecting and debugging facilities. All on conventional hardware—the PC you already use.

Works to develop and deliver your expert systems.

GoldWorks is the only tool that lets you develop and deliver serious expert systems on PC ATs. And **GoldWorks** also takes advantage of PCs based on Intel's powerful 80386 processor, including the COMPAQ DESKPRO 386 and Gold Hill's 386 LISP System.

Works to integrate expert systems with conventional PC applications.

With **GoldWorks,** you can integrate expert system applications with dBASE III and Lotus 1-2-3 . . . integrate C routines and libraries into your expert systems . . . and build and deliver expert systems in network environments.

Works the way you want an expert system builder to work.

GoldWorks from Gold Hill sets the standard by which all other expert system tools will be measured. It was extensively field-tested by developers in dozens of major corporations. And GoldWorks is backed by Gold Hill's comprehensive customer support, training and consulting programs.

Now you have the expert system builder that works the way *you* want to work—**GoldWorks.** To see how it works, order our unique Demonstration Kit, including full color video and complete User's Guide. It's only $49 postpaid, refundable with your **GoldWorks** purchase. To order, call toll-free:

1-800-242-5477.

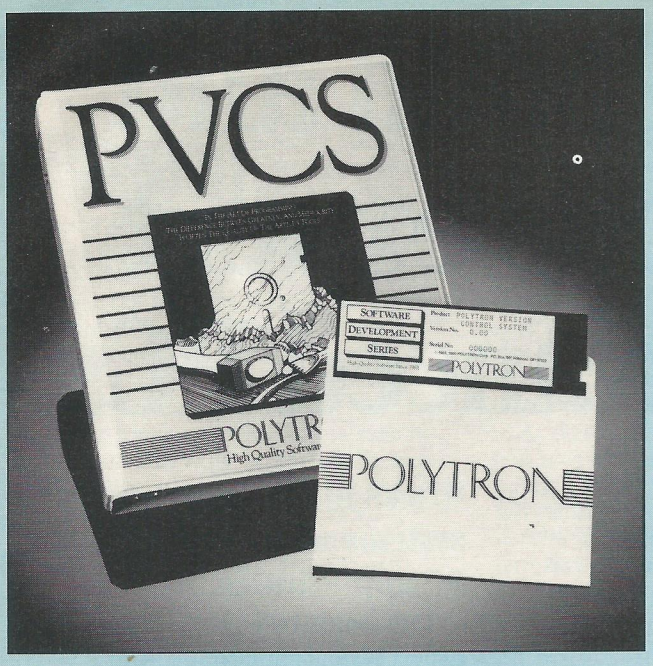
In Mass., call (617) 492-2071.

GoldWorks from Gold Hill. The expert in AI on PCs.

Gold Hill Computers, Inc.
163 Harvard Street
Cambridge, MA 02139

GOLD HILL™

© Copyright 1987, Gold Hill Computers, Inc. Gold Hill, GoldWorks, GCLISP, and 386 LISP System are trademarks of Gold Hill Computers, Inc. San Marco LISP Explorer is a registered trademark of San Marco Associates. Lotus is a registered trademark and 1-2-3 is a trademark of Lotus Development Corporation. dBASE is a trademark of Ashton-Tate. IBM and IBM PC-AT are registered trademarks of International Business Machines Corporation. Intel is a registered trademark and 80286 and 80386 are trademarks of Intel Corporation. COMPAQ and DESKPRO 386 are trademarks of COMPAQ Computer Corporation.



PVCS

THE MOST POWERFUL AND FLEXIBLE SOURCE CODE REVISION & VERSION CONTROL SYSTEM FOR PCS AND LOCAL AREA NETWORKS

PROGRAMMERS, PROGRAMMING TEAMS, PROJECT MANAGERS, LIBRARIANS AND SYSTEM ADMINISTRATORS FIND PVCS INDISPENSABLE

When large and complex software programs and systems are being developed on personal computers, effective management of the revisions and versions becomes critical. The POLYTRON Version Control System (PVCS) simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of PC programmers has enabled us to significantly improve upon these models.

PVCS provides many powerful functions including:

* Storage & Retrieval of multiple revisions of text.
* Maintenance of a complete history of changes.
* Maintenance of separate lines of development using branching.
* Resolution of access conflicts.
* Merging of simultaneous changes.
* Modules can be retrieved by their own revision number, system version number, or specified date.
* Efficient disk storage. PVCS uses a very intelligent difference detection technique that minimizes the amount of disk space required to store a new version.

PVCS Maintains System Integrity

PVCS prevents system corruption that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

The Preferred Version Control System

PVCS is now being used in personal computer software development environments at hundreds of high-technology corporations including:

Hewlett-Packard	3 Com
GTE Data Services	IBM
Sperry	Texas Instruments
Westinghouse Electronics	ROLM Corp.
Nestar Systems	The Aerospace Corp.

Maintains Source Code Written In ANY Language.
Requires DOS 2.0 or higher. Compatible with the IBM PC, XT, AT and other MS-DOS PCs.

Only PVCS meets the needs of Independent Programmers and Corporations. Once you standardize on PVCS, the "Logfiles" used to track and monitor changes are interchangable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced PVCS.

Personal PVCS — Offers most of the power and flexibility of the Corporate PVCS, but excludes the features necessary for multiple-programmer projects. **$149**

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes "Branching" to effectively maintain code when programs evolve on multiple paths (e.g., new versions for different systems, or a new program based on an existing program). Single user. **$395**

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project. 5-Station License $1,000. Call (503) 645-1150 for pricing on Licenses for more than 5 Stations. **$1000**

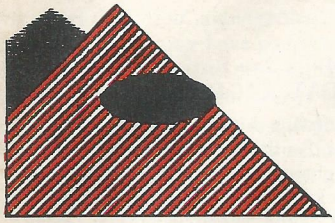
TO ORDER: VISA/MC **1-800-547-4000.** Dept. No. 350. Oregon and outside US call (503) 684-3000. Send Checks, P.O.s to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Dept. 350, Beaverton, OR 97006.

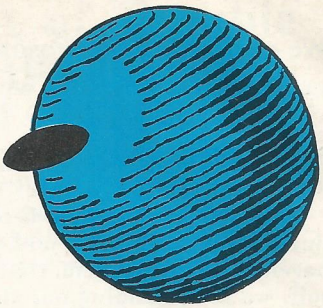
POLYTRON®

High Quality Software Since 1982

CIRCLE 5 ON READER SERVICE CARD

FEEDBACK





Solution for DNA puzzle

Dear Editor:

I read with interest Brian Hayes's "Eight diversions to keep your Cray out of mischief on a cold winter's night" (Theory and Practice, *COMPUTER LANGUAGE*, Feb. 1987, pp. 21-26). It may interest you to know problem 2—create a concordance to the human genome—has been solved.

Two years ago David Lipman and I published a rapid sequence comparison algorithm for proteins ("Rapid and Sensitive Protein Similarity Searches," *Science*, Mar. 22, 1985, pp. 1435-1441). Last year the algorithm was extended to DNA sequences on the IBM PC when floppy disks containing the GENBANK DNA sequence library became available.

It is now possible to compare a newly sequenced clone with the DNA 8,000,000 nt sequence library in less than 3 hours on an IBM PC (or less than 15 minutes on a Sun). As a result, sequencers are creating the concordance as they publish. (I should also mention that soon we will distribute a version of the program that translates the data base and compares DNA sequences at the protein sequence level. This version is substantially more sensitive than DNA sequence comparison but only takes about twice as much time.) We believe the quality of a 3-hour comparison on an IBM PC using this rapid technique is equal to that of a 3-hour comparison on a Cray.

Given the ease of determining the sequence of a protein, your problem 5—structure prediction—becomes much more interesting. But I think many researchers would argue with your premise that "the basic principles that control the folding are straightforward enough." It is only in the last few years that it has been possible to test these principles on synthetic peptides, and some of the results suggest that current principles are greatly oversimplified.

Several years ago another computer magazine had an issue on microcomputers and scientific computing that completely ignored the role of computers in the recent advances in modern molecular and structural biology. I was happy to see that *COMPUTER LANGUAGE* takes a more progressive view of computational problems.

William R. Pearson
Assistant professor of biochemistry
Dept. of Biochemistry
University of Virginia School of Medicine
Charlottesville, Va.

Columnist Brian Hayes responds: Bravo!

Debating curve continuity

Dear Editor:

I read with interest Michael Brannigan's "Generating Curves on the Micro" (*COMPUTER LANGUAGE*, Mar. 1987, pp. 26-32). Unfortunately, his proposed method of curve generation has one limit the user must be aware of, or risk serious consequences.

Certain areas of fluid and solid dynamic design require a piecewise continuous second derivative, while Brannigan'smethod only guarantees a piecewise continuous first derivative.

This is most notable in the area of cam and gear design and in certain aspects of airfoil design. A cam or airfoil drafted using a system (manual or computer) that implemented his method could potentially have disastrous life or separation problems.

Note that the classic cubic spline has a truly continuous first derivative, and a piecewise continuous second derivative. The realization of the need for the condition on the second derivative predated the use of computers to generate such curves. One of the classic mistakes taught to most mechanical engineers in college is the failure of cams with smooth but ill-conditioned surfaces. Airfoils can, under certain circumstances, show similar effects.

On the other hand, Brannigan's method is quite adequate for many applications and is probably very efficient in its use of computer time. I do not wish to discourage potential users from trying this approach. My point is to warn the reader to avoid the use of this approach in the areas of airfoils, cams, gears, and similar dynamic systems.

Walter J. Tingle
Salem, Mass.

Author Michael Brannigan responds: The problem of continuity in the design of curves is important and should be looked at carefully. The ideal situation for most engineering designs is the minimization of the strain energy, or equivalently, the minimization of the mean square curvature as pointed out in the article. The cubic spline does give global continuity of both first and second derivatives. The Bezier curve will give gradient and curvature continuity.

For the method I suggest in the article, if the (λ,s) curve is a natural quadratic spline, then I get minimization of the strain energy with gradient and curvature continuity of the curve. Thus my method in this respect is superior to the spline approach.

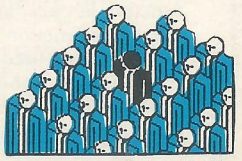
Wrap-up remarks

Dear Editor:

I want to thank your magazine for reviewing numerical programs in Tim Parker's "Math on the micro" (*COMPUTER LANGUAGE*, Apr. 1987, pp. 121-130) and including our product, SolveIT.

There are, however, two issues in the article to which I must respond. The first is the comparison of vaporware with software. Parker reviewed Borland International's Eureka: The Solver, a product that as of this writing (April 1, 1987) has not been released, despite heavy advertising and promotion since November.

I am in favor of preview articles but I feel it was unfair to all of the other competitors in the article to compare the latest unreleased product from Borland with software that was issued over a year ago by the other companies. Our company would gladly have released a preview copy of v.1.4 of our software if we were contacted, and I am sure the other companies would have done the same. The tables and conclusions of the article would have been able to compare their future products.



Artwork: Anne Doering

Microsoft Avoids Challenge

We challenged Microsoft to a C compiler duel-to-the-finish, comparing compile, link and execution times, and we offered to stop advertising for two months if they won...

by Roy Sherrill, President, Datalight

Microsoft purchased our C-compiler during February 1987 and we still haven't heard from them. OK, Microsoft, we are extending our challenge deadline from April 1, 1987 to May 15, 1987. After all, the Microsoft ad claims "the fastest C you've ever seen." Your reply, Microsoft!

Walter says Optimum-C is better

Walter Bright, the developer of Optimum C, says that Optimum C would win 7 out of 10 benchmarks as compared to Microsoft C, V.4.0. Walter explained to me that Optimum C includes a unique global optimizer that helps create compact code while increasing execution speed up to 30%. By the way, Borland, Walter is still waiting for his copy of Turbo C,® V.1.0. Borland's ad claims "the fastest, most efficient and easy-to-use C compiler at any price."

After reviewing Borland's benchmarks, Walter claims that Optimum C is faster. And, as for ease of use, all Datalight C compilers have been shipped with a free Learn C program for the last six months. Also, our new EZ Interactive Editor will show you each syntax error in your source code, then compile or "make" and run your program, all from within the editor. OK, so let the Microsoft challenge begin...

We only ask the following...

The benchmark suite will consist of the set of programs that Microsoft supplied to *Computer Language* for their February 1987 C compiler review issue. Microsoft will make available the programs to Datalight at least two weeks prior to the benchmarking. The benchmarking will be between Microsoft C 4.0 and Optimum-C. It will occur at a mutually agreed upon time and place. Interested individuals will be allowed to attend. The benchmarks will be compiled and run on a standard IBM PC-AT.

There will be two separate tests for each program: compile and link speed, and execution speed. For each test, a representative from each company will set up the compiler so that it performs at its best.

The benchmarks will be adjusted so that they take sufficiently long to run, that the tolerance involved in timing them is insignificant. The winner is determined by the compiler with the faster execution times for the majority of the benchmarks. We'd like an answer from Microsoft no later than May 15, 1987.

So what's a global optimizer?

A global optimizer looks at an entire function at once, analyzing and optimizing the whole function. A technique called data flow analysis is used by Optimum-C to gather information about each function. This enables your compute-bound programs to execute as much as 30% faster after global optimization. But, there is one catch...because the global optimizer ruthlessly searches for ways to speed-up execution speed and minimize memory usage, it has relatively slow compile times. No need to worry, though, because you can merely turn the global optimizer off. In fact, you can select all, none, or part of the following optimizations: constant propagation, copy propagation, dead assignment elimination, dead variable elimination, dead code elimination, do register optimizations, global common subexpression elimination, loop invariant removal, loop induction variables, optimize for space, optimize for time, and very busy expressions.

Choose from five memory models

Speed your programs by selecting the memory model that best suits your application.

Model	Code	Data
Compact	64k total code & data	
Small	64k	64k
Program	1M	64k
Data	64k	1M
Large	1M	1M

Compiling, one step...

Now with the one step DLC program you can create .OBJ, .EXE and .COM files. Also, DLC can handle multiple files and run MASM on your assembly files.

Try Optimum-C risk free

Try Optimum-C for 30 days and if you are not 100% satisfied return it for a full refund. Also available is a C tutorial which is a combination workbook and floppy disk to help lead you through the C language with tutorials, quizzes, and program exercises.

O.K. Microsoft, it's up to you. We've put two months of advertising on the line that says you can't beat Optimum-C to a real test. Your answer, please?

PRICES

Developer's Kit w/ C Tutorial	$ 99
Optimum-C w/ C Tutorial (both with library source)	$139

Add $7 for shipping in US/$20 outside US
COD (add $2.50)

Not Copy Protected

ORDER TOLL-FREE TODAY!

1-800-221-6630

ATTENTION OEMs!

Contact us regarding arrangements.

Microsoft and MS-DOS are registered trademarks of the Microsoft Corporation. Turbo C is a registered trademark of Borland International.

Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBBS, August 1986

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

COMPUTER LANGUAGE, February 1986

Optimum-C Version 3.0

NEW!
EZ Interactive Development Environment
NEW!
Inline 8087/80287 Math Support

- Full UNIX System 5 C language plus ANSI extensions
- Fast/tight code via powerful optimizations including common sub-expression elimination
- DLC one-step compile/link program
- Multiple memory model support
- UNIX compatible library with PC functions
- Compatible with DOS linker and assembler
- Third-party library support
- Automatic generation of .COM files
- Supports DOS pathnames, wild cards, and Input/Output redirection
- Compatible with Lattice C version 3.x
- Interrupt handling in C
- Debugger support
- ROMable code support/start-up source

MS-DOS® Support Features

- Mouse support
- Sound support
- Fast screen I/O
- Interrupt handler

MAKE Maintenance Utility

- Macro definition support
- MS-DOS internal commands
- Inference rule support
- TOUCH date manager

Tools in Source Code

- cat—UNIX style "type"
- diff—Text file differences
- fgrep—fast text search
- pr—Page printer
- pwd—Print working directory
- wc—Word count

Datalight

17505-68th Avenue NE, Suite 304
Bothell, Washington 98011 USA
(206) 367-1803

CIRCLE 6 ON READER SERVICE CARD

The second point is that there was a bias by the reviewer in using MathCAD "as the standard against which the other packages were judged." That would have been fine if the other packages were built as MathCAD clones. Two of the packages, muMATH and SolveIT, are not designed to attack problems in the same way that MathCAD does and, consequently, these two products were lacking in the summary tables because they did not fit the MathCAD mold.

It was decided not to use the free format entry form of MathCAD for SolveIT because in our experience it emphasizes typing skills while limiting the depth of problems that can be solved. Instead, a context-sensitive prompting screen was chosen requiring only a few keystrokes to invoke powerful functions and build complicated macros. This interface was made to look like an RPN calculator to keep it familiar to the millions of scientists, engineers, and students who do not want to be systems programmers. As Parker pointed out, our manual "seems ideal for a novice computer user" and we want it to be used for engineering and science whether or not the users are experienced programmers.

As for the table, for our present product we should have "yes" for differentiation, differential equations, statistical functions, complex variables, graphics, and data I/O from files. We will continue to produce a unique and rapidly improving product.

Thomas L. Ochs, president
Structured Scientific Software Inc.
Bend, Ore.

Dear Editor:

I found Tim Parker's review "Math on the micro" a quite interesting and very helpful summary of microcomputer numerical analysis programs. His review made me realize that the state of the art of these programs is considerably advanced from what I had envisioned. A program like Mathsoft's MathCAD would appear to be nearly indispensible to working engineers or physicists at the least and useful in many other fields.

However, Parker's review has a very serious defect. Although his comments about Microsoft's muMATH are valid within the context of numerical analysis, muMATH was designed for very different kinds of work, and not to stress that does an injustice to a very fine piece of programming. muMATH was designed for symbolic, rather than numerical, mathematics. Denigrating its numerical capabilities is like dismissing the capabilities of a microcomputer because it is awkward to use as a calculator.

Clearly muMATH, not MathCAD, is the most unique of the packages reviewed. None of the others can, for example, symbolically simplify trigonometric expressions or verify trigonometric identities for you. While the other programs may be able to invert matrices whose elements are given as numbers, only muMATH can do the same when one or more of the elements are algebraic symbols. The others may be able to numerically solve a simple, ordinary differential equation with numerical coefficients; muMATH can often give you the general solution to that same equation.

For example, consider a problem that recently arose in a mechanics class I am teaching. One end of a string is tied to a fixed cylinder and the other to a ball. The ball is thrown in such a way that the string wraps around the cylinder, causing the ball to spiral inward as it moves. The differential equation for the angular displacement (A) of the ball and string (length L) around the cylinder (radius r) is (t is the time):

$$(L - rA)d^2A/dt^2 - r(dA/dt)^2 = 0.$$

I would wager that none of the other programs reviewed can give accurate solutions for $A(t)$ over the full range of motion, even if you choose specific values for L and r. The reason is that the speed of the ball becomes very large as A approaches L/r. Ask muMATH for the solution and you get the general solution, which is:

$$A(t) = L/r \pm \mathrm{sqrt}(2art - 2br + L^2)/r.$$

(The constants a and b are integration constants determined by the initial conditions.)

Admittedly, it is fortuitous that there is an analytic solution for this problem, but the fact remains that I would not have spent the time trying to find the solution without muMATH or a similar symbolic manipulation program. (I was surprised such a solution exists.)

muMATH's capabilities run far beyond what I can even briefly mention in this letter. Suffice it to say that it can accurately perform algebra, trigonometry, calculus, matrix, and vector operations at a level exceeding typical introductory physics or engineering majors. Because it can deal so well with tedious algebra, I have found it exceedingly useful in my teaching and research, even in the short time I have had it (about a month).

My point is not that muMATH is the best of the programs reviewed, but that its abilities run in a completely different direction than those of the other programs, making it a good complement to one of the numerical analysis packages. Far from "beginning to show its age," muMATH remains (so far as I know) the only commercial symbolic manipula-

C & PASCAL PROGRAMMERS

Blaise Computing provides a broad range of programming tools for Pascal and C programmers with libraries designed for serious software development. You get carefully crafted code that can be easily modified to grow with your changing needs. Our packages are shipped complete with comprehensive manuals, sample programs and source code.

C TOOLS PLUS
$175.00
NEW! Full spectrum of general-purpose utility functions; windows that can be stacked, removed, and accept user input; interrupt service routines for resident applications; screen handling including EGA 43-line text mode support and direct screen access; string functions; and DOS file handling.

PASCAL TOOLS/TOOLS 2
$175.00
Expanded string and screen handling; graphics routines; easy creation of program interfaces; memory management; general program control; and DOS file support.

VIEW MANAGER
$275.00
Complete screen management; paint data entry screens; screens can be managed by your application program; block mode data entry or field-by-field control. Specify C or IBM/MS-Pascal.

ASYNCH MANAGER
$175.00
Full featured asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; baud rates up to 9600; modem control and XMODEM file transfer. Specify C or IBM/MS-Pascal.

Turbo POWER TOOLS PLUS
$99.95
NEW! Expanded string support; extended screen and window management including EGA support; pop-up menus; memory management; execute any program from within Turbo Pascal; interrupt service routine support allowing you to write memory resident programs; schedulable intervention code.

Turbo ASYNCH PLUS
$99.95
Complete asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; and baud rates up to 9600.

RUNOFF
$49.95
NEW! Text formatter written especially for programmers; flexible printer control; user-defined variables; index generation; and general macro facility. Crafted in Turbo Pascal.

EXEC
$95.00
Program chaining executive. Chain one program from another even if the programs are in different languages. Shared data areas can be specified.

ORDER TOLL-FREE 800-227-8087!

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 7 ON READER SERVICE CARD

tion program capable of running on a microcomputer.

Ronald L. Greene
Dept. Of Physics
University of New Orleans
New Orleans, La.

Dear Editor:

The review of a collection of mathematics software packages, "Math on the micro" was quite unfair in its assessment of muMATH. In effect, the reviewer lumped a racehorse with a collection of dairy cattle, and he didn't like the quality of the milk it gave.

All the packages reviewed, except muMATH, are engineering tools, generally intended for the approximate, numerical solution of equations or systems of equations. muMATH, in contrast, is designed for problems involving exact symbolic computation. I get the impression that your reviewer doesn't clearly appreciate the differences between these two categories of problems. To get some feeling for the differences, he should try using the various packages to invert a matrix with nonnumeric entries, or perhaps to calculate the resultant of two fifth degree polynomials with symbolic coefficients. He might then have a different opinion about which of the programs is able to carry through "a complex sequence of calculations."

Richard Stauduhar
Berkeley, Calif.

We've always debated at COMPUTER LANGUAGE *whether or not to review beta or prerelease copies of major new products. Our general policy is to wait until we can review a production copy. We did not do that with Borland's Eureka: The Solver because of the high interest among our readership in the product.*

In retrospect, however, that may have been a mistake. First, as Thomas Ochs points out, it is unfair when comparing products. Second, it can be a disservice to our readers to review a product that can't yet be bought and may be different when it appears on the market than what was originally reviewed.

In the future, we plan to stick more closely to our policy of only reviewing production copies and, in particular, not doing a beta version review when comparing products.—Ed.

Reviewer Tim Parker responds: The comments regarding muMATH are well taken, but the basis for my review was numerical analysis. muMATH was evaluated in that respect although I tried to point out the symbolic capabilities of the package. The first sentence of the second paragraph specifically states that muMATH is "an interactive symbolic math" system. I hope I did not give the impression I was dismissing the package because of its symbolic mathematics potential.

I have been a muMATH user for almost six years now, both CP/M and MS-DOS. It is a fine product for symbolic mathematics.

Mixed reviews for Leary

Dear Editor:

Shame on you! First you deny that your April issue contains humor (Publisher's Notes, *COMPUTER LANGUAGE*, Apr. 1987, p. 4). Then, 107 pages later, you have that hilarious interview "Timothy Leary and the CyberPunks" (ComputerVisions, *COMPUTER LANGUAGE*, Apr. 1987, pp. 111-116). Although I enjoyed your April Fool's issue last year, I must say that this year's inclusion of Leary is even better.

After dismissing the canard that he is "burned out," "toasted," "a babbling idiot," "strung out, spun out, freaked out, or blitzed out," Leary then shows us the danger of technobabble: it is possible to fill four pages with computer jargon yet say almost nothing concrete.

In the Good Fight against Big Brother's dumbing-down of America, we are indeed fortunate to have the far-seeing Timothy Leary on our side.

Steve Snyder
Tulsa, Okla.

Dear Editor:

In "Timothy Leary and the CyberPunks" Leary states that computer programmers have a "great reponsibility and a wonderful challenge" of raising the intelligence level of the human race by 10%. This is one of the most ridiculous statements I have read anywhere in a long time. However, considering that you were interviewing someone who admits he still uses drugs for amusement, I am not surprised.

Please consider the content of your articles more carefully in the future. Some of your readers may actually take something like that seriously. I will personally stick to trying to save people time and money by leaving intelligence in the hands of our Creator, who certainly has not delegated the responsibility for changing it to mere men! Thank you.

Bob Nance
NEWLIFE Software
Clarksdale, Miss.

Welcoming WHEREFOR

Dear Editor:

Good job with the April Fool's issue. I'm glad you gritted your teeth and retained an element of balderdash, considering the abuse you received last year.

I was delighted to see my favorite language, WHEREFOR, spotlighted in Jeffrey Star's "WHEREFOR: Destroying another myth" (Exotic Language of the Month Club, *COMPUTER LANGUAGE*, Apr. 1987, pp. 103-104). We have an extensive library of WHEREFOR support routines that run on an IBM 3090-400 mainframe, but they are not all machine dependent and many could be ported to micros.

Our most loved WHEREFOR routine is *demo(level)*. The parameter to demo is the level of importance of the demonstration you will soon be giving. Demo will branch to an undisclosed location whenever the person receiving the demonstration asks a "can it do this" question.

Another standby is *io*. *io* reads a 128-byte file on disk and returns several minutes later. It seems to fail on a 4.77 MHz PC with a floppy disk since the PC cannot adequately emulate a true mainframe bottleneck.

The routine we use most often is an adaptation of WHEREFOR's *if motivated* case structure. *If critical* uses artificial intellligence to determine the optimum time to do something unexpected. It will usually call WHEREFOR's *die* when you are on the verge of isolating a particularly difficult bug. IBM has written a version of *if critical* called OS/MVS, which does basically the same thing but has a more complicated interface and requires frequent user intervention.

You are always welcome to any of our WHEREFOR routines, including *randomerror* and *if Friday*. Meanwhile, keep up the good work.

Shelton Alsup
Omaha, Neb.

Faulty fractals

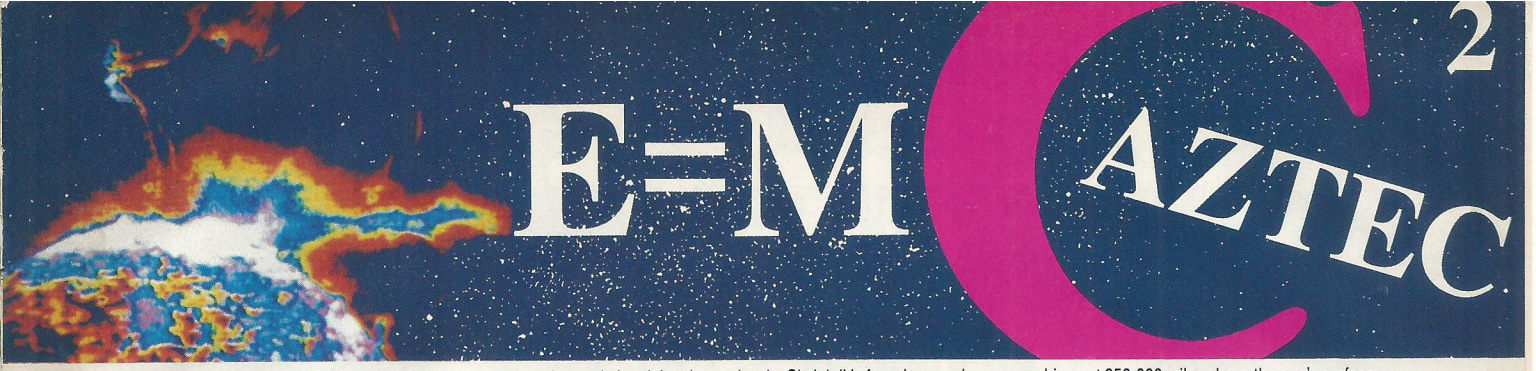
Dear Editor:

In my article "Formulating Fractals" (*COMPUTER LANGUAGE*, Apr. 1987, pp.28-40), the following typographical errors occurred:

- The head on Figure 4D should be "Julia set of $f(z)=z^2+i$."
- On page 34 beginning at line 47 the text should read: "It is preserved under backward iteration, $(f^{-1})_n$. For example, if:

$$w = f(z) = z^2 + \lambda$$

$$z = f^{-1}(w) = \pm \sqrt{w - \lambda}$$



Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

Genius Begins With A Great Idea ...

But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple // C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

Aztec C86 4.1
New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p Professional System $199
• optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System $299
• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System. $499
• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: Essential Graphics • C Essentials • C Utility Library • Greenleaf Com. • Greenleaf General • Halo • Panel • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data C terp • db Vista • Phact • Plink86Plus • C-tree.

CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c (CP/M-80 & ROM). $349
Aztec CII-d (CP/M-80) $199
Aztec C80 (TRS-80 3&4) $199

Aztec C68k/Am 3.4
New Amiga Release

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impressive.

Aztec C68k/Am-p Professional $199
A price/feature/performance miracle. System includes: optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Amiga libraries • examples.

Aztec C68k/Am-d Developer $299
The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

Aztec C68k/Am-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C68k/Mac 3.4
New Macintosh Release

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

Aztec C68k/Mac-p Professional $199
• optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Macintosh libraries • examples.

Aztec C68k/Mac-d Developer. $299
The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

Aztec C68k/Mac-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C65
New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

Aztec C65-c Commercial $299
• runs under ProDOS • code for ProDOS or DOS 3.3

Aztec C65-d Developer. $199
• runs under DOS 3.3 • code for DOS 3.3

Aztec ROM Systems
6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under $1,000 outperform systems priced at over $10,000.

Initial Host Plus Target $750
Additional Targets $500
ROM Support Package $500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

C' Prime
PC/MS-DOS • Macintosh
Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as $500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime $75

Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus $10. Site licenses, OEM, educational, and multiple copy discounts are available.

MANX

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

To order or for more information call today.

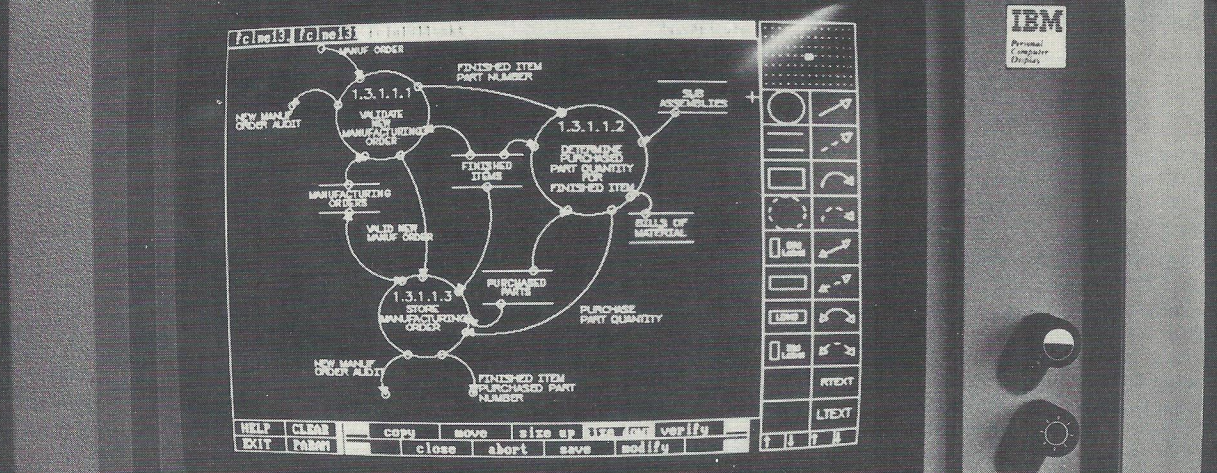
1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

MS is a registered TM of Microsoft, Inc., CP/M TM DRI, HALO TM Media Cybernetics, PANEL TM Roundhill Computer Systems, Ltd., PHACT TM PHACT Assoc., PRE-C, Plink-86, Plink-86+, P-Force TM Phoenix, db Vista TM Raima Corp., C-terp, PC-lint, TM Gimpel Software, C-tree TM Faircom, Inc., Windows for C, Windows for DATA TM Creative Solutions, Apple II, Macintosh TM Apple, Inc., TRS-80 TM Radio Shack, Amiga TM Commodore Int'l., Unix TM AT&T, Vax TM DEC, Aztec TM Manx Software Systems.

CIRCLE 8 ON READER SERVICE CARD

INTELLIGENT CASE SOFTWARE AT AN INTELLIGENT PRICE.



Intelligent Systems Analysis begins with intelligent software choices.
Meet your most intelligent choice.
The Yourdon Analyst/Designer Toolkit.
At $1995, the Analyst/Designer Toolkit has everything you need to support all of your MIS projects, no matter how complex; the most advanced graphics system of any PC-based CASE tool lets you create all of the diagrams associated with the Yourdon Structured Techniques, raises your productivity levels, and increases your work speed by as much as 30%.

$1995

Error and consistency checking, an integrated data dictionary, and one of the easiest to use menu systems make for a professional system that offers everything you could want in a CASE system—at a remarkably low price.
And lest we forget: the Toolkit includes technical support from the people who know Structured Analysis and Design better than anyone. And we ought to.
After all, we wrote the book on it.
Now we've written the software.

The Yourdon Analyst/Designer Toolkit.™

YOURDON, INC. A Subsidiary of DeVRY, INC. 1501 Broadway New York, N.Y. 10036 TEL. (212) 391-2828

CIRCLE 9 ON READER SERVICE CARD

iterating:

$w_{n+1} = f^{-1}(w_n) = (f^{-1})_n(w)$."

•On page 35 beginning on line 48 the text should read: "(for example, using $|z|^2$ instead of $|z|$, thus eliminating a square root in each iteration)."

The dynamics for the color photos are:

Pages 28,29,30: $f(z) = z^2 + \lambda$, parameter space.

Page 33: $f(z) = \lambda - \exp(z)$, parameter space.

Cover, page 35: Newton's method applied to $f(z) = z^7 - 1$.

Pages 37,40: $f(z) = z^4 - z - \lambda$, parameter space.

I also forgot to define some parameters in NEWT. This program calculates pixel information for a rectangular region R. The parameters are:

Cols, rows. Pixel resolution of screen.

Acorner. Real part of lower left hand corner of R.

Bcorner. Imaginary part of lower left hand corner of R.

Side. Length of the horizontal side of R.

Iterations. Maximum number of times the calculation loop was executed.

For a list of these parameters for the color pictures, see Table 1.

Iteration is by no means limited to quadratics. I've had good success with many other functions, such as:

$f(z) = \lambda \tan(\lambda z)$
$f(z) = \arcsin(\frac{z}{\lambda})$

(Here, one must be careful with scaling, For example, exponential growth to logarithmic scaling.)

Fractals are quite raw—there's certainly a bit left to explore.

Steve Casey
Silver Spring, Md.

Clarifications

In Mike Higgins' "Advanced MS-DOS Batch Procedures" (COMPUTER LANGUAGE, *Apr. 1987, pp. 45-49), the line to increase environment space on an MS-DOS 3.xx system should be:*

```
SHELL=C:\COMMAND.COM C:\
  /P/E:1024
```

*The code in T.A. Elkins' review of APL*PLUS PC System v.6.2* (COMPUTER LANGUAGE, *Apr. 1987, pp. 135-138) was typeset incorrectly on the first page. The three example outputs should have been:*

```
U.S.A.
   Japan
   China

U.S.A.
   Japan
   Hongk

U.S.A.
   Hongk
ong Ja
```

The table "Integration, delivery, and support" in the software review "The getting of wisdom: PC expert system shells" (COMPUTER LANGUAGE, *Mar. 1987, pp. 117-150) had two incorrect entries for IntelligenceWare's Intelligent/Compiler. According to the company, the entries for telephone support and consulting should have been "yes."*

Input parameters for color photographs

Newton's method for f(z)=z7−1**

Acorner	Bcorner	Side	Iterations
−.75	.25	.6	64

f(z)=z2+l Mandelbrot set (parameter space)**

Acorner	Bcorner	Side	Iterations
−.748	.106	.0028	1024

z2+l Mandelbrot set (parameter space)**

Acorner	Bcorner	Side	Iterations
−.74547	.11298	.000085	1024

f(z)=l*exp(z) (parameter space)

Acorner	Bcorner	Side	Iterations
−.08	3.08	.31	15

f(z)=z4−z−l (parameter space)**

Acorner	Bcorner	Side	Iterations
−.8222	.3712	.0042	1024

Table 1.

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2.** C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for $195:
- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

To order:
send check or money order to:

GUIDELINES SOFTWARE
P.O. Box 749
Orinda, CA 94563

To order with Visa or MC, phone (415) 254-9393.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

CIRCLE 10 ON READER SERVICE CARD

Version 3.0

New Features

Windows, Data Entry, Help Management, Menus, Text Editing, plus ...

SOURCE CODE

FREE

Vitamin C

It's good for your system!

The Vitamin C Difference

With **Vitamin C,** your applications come alive with windows that explode into view! Data entry windows and menus become a snap, and context sensitive pop-up help messages are nearly automatic.

With **VCScreen,** you'll save time by interactively painting windows and forms so what you see is what you get! Then, one button generates C source code ready to plug into your program and link with Vitamin C.

Easy enough for the beginner. Versatile enough for the professional. Vitamin C's **open-ended design** is full of "hooks" so you can intercept and "plug-in" special handlers to customize or add features to most routines.

Of course, Vitamin C **includes all source code FREE,** with no hidden charges. *It always has.* That means you'll have everything you need to adapt to special needs without spending hundreds of dollars more.

Windows

Create as many windows as you like with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, pop-up, pull-down, zoom-in, 4-way scrolling, scroll bars, sizes up to 32k, text file display & editing, cursor display, and more.

Unique built-in feature lets users move and resize windows during run-time via a definable key.

Access the current window by default or a specific window any time, even if it's hidden or invisible. Save and load windows on disk for more versatility!

Data Entry

Flexible dBase-like data entry and display routines feature protected, invisible, required, and scrolling fields. Picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited data validation via standard and user definable routines. That means you aren't locked into one way of doing things.

Vitamin C even provides true right-to-left input of numeric fields with dynamic display of separators & currency symbols.

High Level Functions

Use our intergrated help management, multi-level menus, and text file routines, or build your own handlers using Vitamin C's basic windowing and data entry routines.

Standard help handler provides context sensitive pop-up help messages any time the program awaits key strokes. The help text file is stored on disk and indexed for quick access. So easy to use that a single function initializes & services requests by opening a window, locating, formatting, displaying, and paging through the message.

Multi-level "MacIntosh" & "Lotus" style menus make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens, quickly and easily.

Text editor windows can be opened for pop-up note pads, memo fields, or general purpose editing. Features include insert, delete, word wrap, and paragraph formatting.

VCScreen

Screen Painter/Code Generator

Just as Vitamin C's reusable functions speed your programming, VCSreen makes it even *faster and easier* by automatically generating C source code for your data entry screens!

With VCScreen's interactive screen editor, you actually draw your forms. You can define input, output and constant fields, headings, boxes, lines and even a window for the form to run in.

What you see is what you get. If you don't like the position of an object, just "pick it up" with the cursor and move it! Changing colors, attributes, copying, and deleting is just as easy.

VCScreen generates readable C source code. It declares variables with names you provide and can even generate structures.

With VCScreen choosing the right functions, parameters and sequences, and Vitamin C supplying the functions to choose from, you can stop worrying about semi-colons, matching braces, and calling conventions and concentrate on creating your application!

30 Day Money Back Guarantee

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

Vitamin C $225.00

Includes ready to use libraries, tutorial, reference manual, demo, sample, and example programs, and quick reference card. For IBM PC and compatibles. Specify Microsoft, Lattice, Computer Innovations, Aztec, Mark Williams, Wizard, DeSmet, or Datalight C compiler AND compiler version number when ordering.

Vitamin C Source ... FREE*

**Free with purchase of Vitamin C*

VCScreen $99.95

Requires Vitamin C and IBM PC/XT/AT or true compatible.

ALL ORDERS:

SHIPPING: $3 ground, $6 2-day air, $20 overnight, $30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on a U.S. Bank. Texas residents add 7¼% sales tax.

For Orders or More Information, Call ...

(214) 416-6447

creative PROGRAMMING

Creative Programming Consultants, Inc.
Box 112097 Carrollton, Texas 75011

PROGRAMMING ON PURPOSE

Divorcing data structures

By P.J. Plauger

Last month I introduced outside-in design, which focuses on the problem of structuring modules that have multiple nontrivial data structures as input or output. The examples I gave all involved marrying the multiple structures from the outside to form a composite data structure that shaped the code inside the module. Thus the name outside-in design.

This month I continue the discussion of outside-in design with a hard look at the many cases in which such a marriage of data structures doesn't work. To say that the marriage doesn't work is not an absolute statement. Rather, it involves a judgement that the resulting structure is so complex it doesn't help control the complexity of the problem to be solved. It adds to it.

So what do you do? The answer is best summarized in a Brian Savage cartoon I saw several years ago. (I am a sucker for a well-aimed cartoon, and Savage often lives up to his name.) It shows a marriage counselor addressing a young couple, who are obviously thunderstruck. He is saying, "What's the big deal? If you can't get along, why not get a divorce?" That's all there is to it.

In programming terms, you must put some distance between the various data streams that don't fit together well. Where originally you had a single module, replace it with two or more. Each of the new modules inherits just one of the troublesome data streams. Naturally, you must connect these multiple modules by introducing new data streams. Otherwise they could not possibly cooperate to perform the task specified for the original module. The real design work comes in specifying the structure of the newly introduced streams.

What you need to do is determine a simpler structure that is common to the more complex structures. You want this structure to be as complex as possible, to stay close to the original data structure, yet simple enough to marry nicely with two or more of the more complex structures you started out with. You see, you can't get away from the necessity of marrying all of the data structures for streams that enter or leave a single module. All you can do is break up a difficult problem into two or more simpler problems.

Figure 1 shows a simple example in which input and output data structures clash. The clash is illustrated symbolically by an input structure that repeats in groups of threes and an output structure that repeats in groups of twos. Since 2 and 3 are relatively prime numbers, the simplest structure that accommodates both is a repetition of groups of sixes. (Please be sure to write if I got that math wrong.) That's not an awful data structure, but it's certainly not a compelling design to repeat any code in six variations. And as with all combinatorics, things get bad very fast with more factors.

Figure 2 shows the obvious way to resolve this structure clash. You break the original module into two, and connect the two with a stream whose structure is a simple repetition of the common atomic structure. Now you have two simpler problems.

The left module in the data stream is a classic input driven form, reducible by left-to-right design. It is a parser that must produce single records as they are recognized. (See "Recognizing input," Parts I and II *COMPUTER LANGUAGE*, Nov. 1986, pp. 15-19, and Dec. 1986, pp. 13-16.) The right module in the data stream is a classic output driven form, reducible by right-to-left design. A data generator, or report writer, must be used to compose more complex forms from a stream of single input records. (See "Generating data," *COMPUTER LANGUAGE*, Sept. 1986, pp. 15-19.)

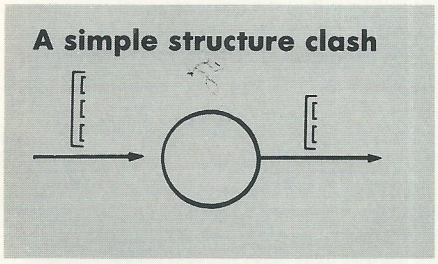


Figure 1.

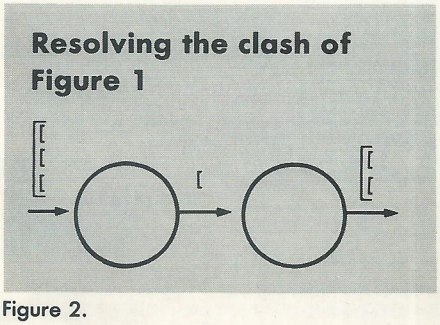


Figure 2.

For a more concrete example, consider a text formatter. Shorn of its whistles and bells, a text formatter spends much of its time reading lines of text written any old way and composing them into lines of more or less uniform width. In the absence of any hyphenation algorithms, output lines are split between words. You start a new line after you have placed at least one word on a line and before you place a word that will make the line too long. With this simple rule, and a fairly moronic definition of what constitutes a word, you can do a remarkably pretty job of formatting text.

The structure clash is obvious. There is no predeterminable relationship between the words presented on any one input line and the output lines they generate. A given input line could complete one line, generate several more whole lines, and/or provide the beginning of yet another. I have seen programs that nevertheless read lines, unpack them into words, repack words into output lines, and put out the lines, all in one giant control loop. What such programs lack in aesthetics they make up for in job security for the original authors.

Resolving the structure clash is equally obvious. You want an input module that reads lines of text and delivers separate words. You also want an output module that eats words one at a time and produces lines to be written out. The intermediate data stream you introduce is a sequence of zero or more words.

Of course, you could also introduce an

intermediate data stream that consists of individual characters. In a UNIX-like environment in which each of the modules is a separate program and the two are connected by a pipeline, this is a compelling solution for the intermediate stream. But that is mostly because UNIX traffics heavily in unstructured streams of individual characters. You always have more machinery to help you if you adopt the local lingua franca.

If both modules are within the same program, however, it may be harder to justify introducing a character stream. I am assuming that there are ways in which the line structure of the input affects how you break the input into words. (You may tolerate hyphenated words split across input lines, for instance.) In that case, you may find yourself recognizing words in the input module and then decomposing them for transmission. Meanwhile, the output module is reconstituting words from the connecting data stream so that they can be properly packed into output lines. This is makework.

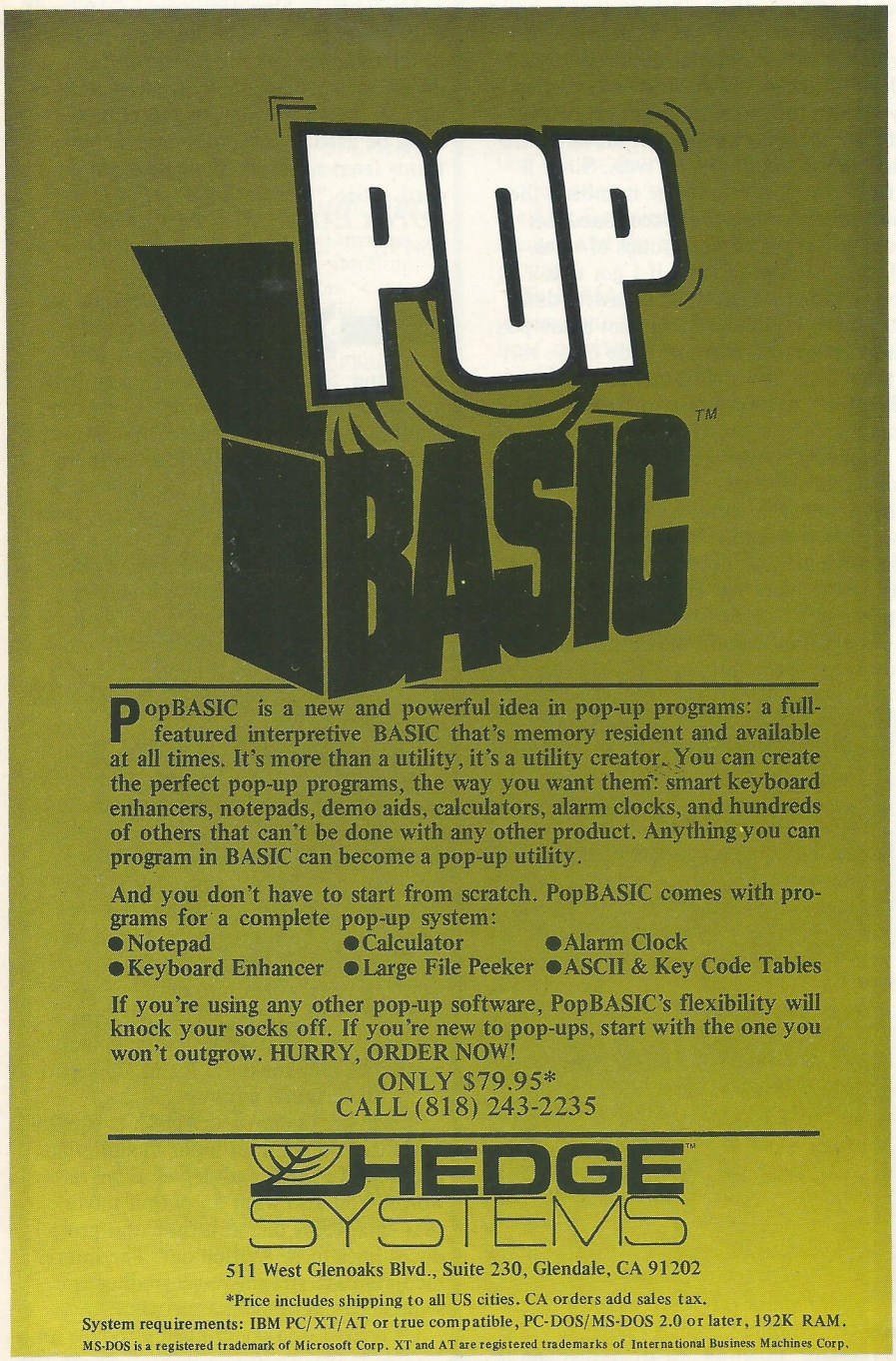
And even if you don't object to communicating via streams of characters, you would probably balk at communicating via bit streams. Everyone draws the line somewhere.

Examples abound, in fact, in which input data structure clashes with output data structure. Disk files are almost invariably composed of a sequence of fixed-size blocks. If a file contains variable length records, be they text lines or more structured entities, it is commonplace for records to span block boundaries. You have a clash just separating the record structure from the block structure.

A program that reads one disk file and writes another faces multiple clashes. It must read disk blocks, compose input records, perform its principal business of transforming input records to output records, then sprinkle output records across a different pattern of disk block boundaries. Small wonder that most of us lean heavily, and gratefully, upon operating systems and I/O libraries to handle most of this business for us under the hood.

A programmer's vocabulary is replete with jargon for various stylized ways of resolving structure clashes. We speak of blocking and deblocking, packing and unpacking, buffering and unbuffering. Think of all the programs that compose screens, paginate output, or assemble packets for transmission. Each is isolating an output stream with nontrivial data structure from an input stream which has an unrelated data structure that is also nontrivial.

Structure clashes do not occur only when input structure meets different output structure. Figure 3 shows a stylized example of a module that accepts relatively unstructured input but must produce two output streams, each with its own peculiar



CIRCLE 11 ON READER SERVICE CARD

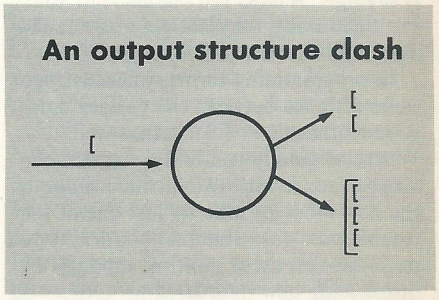


Figure 3.

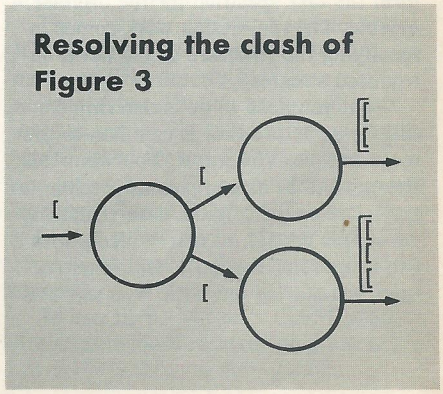


Figure 4.

WINDOWS FOR DATA™

The first choice of professional C programmers

"Windows for Data is the best programming tool I've ever used. It's the most flexible I've seen. Whenever I've wanted to do something, I've been able to find a way."

Steven Weiss,
Stratford Systems

Professionals choose our tools because they are designed, crafted, and supported for professionals. Here at Vermont Creative Software, we understand that performance and pleasure in programming derive from more than a long list of functions. **Windows for Data** provides:

PROFESSIONAL FLEXIBILITY: Our customers repeatedly tell us how they've used WFD in ways we never imagined - but which we anticipated by designing WFD for unprecedented adaptability. Virtually every capability and feature can be modified to meet special needs. You will be amazed at what you can do with WFD.

PROFESSIONAL PERFORMANCE: Screen output is crisp and fast. Windows, menus, and data-entry forms snap up and down from the screen. WFD is built upon and includes **Windows for C, the windowing system rated #1 in speed and overall quality** in PC Tech Journal (William Hunt, July 1985).

PROFESSIONAL RELIABILITY: An unreliable tool is worse than no tool at all. VCS products are known in the industry for their exceptional reliability. Ask anyone who owns one.

PROFESSIONAL DOCUMENTATION: Over 600 pages of documentation provide step-by-step explanations for each major application, a reference page for each function, listings of functions alphabetically and by usage, and a fully cross-referenced index. Extensive tutorials and demonstration programs assist learning.

PROFESSIONAL TECHNICAL SUPPORT: The same expert programmers that develop our products provide prompt, knowledgeable technical support.

PROFESSIONAL PORTABILITY: High-performance versions of VCS products are available for XENIX, UNIX, and VMS, as well as DOS. No royalties on end-user applications.

OUR CHALLENGE AND GUARANTEE

If you have an application where no other tool can do the job, try **Windows for Data.** If it doesn't help you solve your problem, RETURN FOR A FULL REFUND. YOU MUST BE SATISFIED.

Ask for **FREE DEMO DISKETTE**

Vermont Creative Software

21 Elm Ave.
Richford, VT 05476
Telex: 510-601-4160 VCSOFT
Tel.: 802-848-7731

Prices: PCDOS* $395; XENIX, VMS, UNIX Call.
*PCDOS specify C compiler.

WINDOWS FOR DATA

for DOS, UNIX, VMS ...

The complete windowing data entry, menu, and help system that does the hard job others can't — we **guarantee** it!

Pop-up data entry windows; field types for all C data types, plus decimals, dates, and times; auto conversion to and from strings for all field types; system and user supplied validation functions; range checking; required, must-fill, and protected fields; free-form movement; multiple-choice field entry; scrollable sub-forms. Branch and nest windows, forms, and menus.

Complete context-sentitive help system with pop-up windows and scrollable text.

Pop-up, pull-down, scrollable, and Lotus-style menus.

NEW FOR DEBUGGING: Exclusive **VCS Error Traceback System** automatically identifies the location and cause of program errors. Eliminates the need to code error checks on all function calls! **VCS Memory Integrity Checking** helps catch those hard-to-detect, memory-corruption errors.

NEW FOR ERROR HANDLING: Install your own error handler to be called whenever a function detects an error.

NEW FORM LAYOUT UTILITY simplifies form design.

CIRCLE 12 ON READER SERVICE CARD

structure. If the two output structures do not marry easily, you have just as much trouble arriving at a clean data-structured organization of the module as when input and output structures clash.

Figure 4 shows a data flow you can introduce to resolve the output structure clash of Figure 3. This time we add two modules because the original module still has plenty of work to do in splitting the data stream. It is generally best not to pollute it with the problem of handling one of the nontrivial output structures as well.

Splitting modules this way is just another application of the principle of information hiding. D.L. Parnas deserves credit for expressing this principle most clearly, in a number of sermons. Each module is responsible for the detailed knowledge of one (and preferably just one) data stream. Ideally, everyone else calls upon one module to generate data for that stream and upon a companion module to consume data from that stream. Information about the structure of the stream is shared between these two modules but hidden from the rest of the universe. That way, changes in the structure of the stream can safely be effected by changing just the producer and consumer modules—nothing else should be perturbed by the change.

The price you pay for working with more tractable modules, of course, is that you have more of them. You must be sure that, given your implementation of data streams and modules, you do not lose too much performance for the elegance you gain. So let's talk about implementations for awhile.

Systems analysts have learned to communicate as much as possible through drawings with neat circles connected by arrows. This is the most abstract way of representing the flow of data and the transformations that must occur upon that data to get the job done. You have maximum freedom to understand and refine the fundamental data flow at this abstract level.

In fact, the job of the systems analyst can be summarized neatly in four steps:

1. Determine the current physical data flow in the system under study. Draw a physical data flow diagram, showing paper flow, phone lines, etc.

2. Determine the current logical data flow. Redraw the physical data flow diagram in terms of abstract data streams and transforms.

3. Determine the desired logical data flow. Alter the current logical data flow diagram to simplify communications, add new functions, resolve structure clashes, etc.

4. Determine the desired physical data flow of the improved system. Draw a new physical data flow diagram, showing implementation decisions for the desired data flow diagram.

We tend to take for granted that the result of a systems analysis will be a computer program (or suite of programs) that performs all of the functions identified in the analysis. A good analyst will keep in mind that parts of a system can be implemented in many different ways that are better handled outside the computer. (A very good analyst will keep open the possibility that the entire system might be better handled outside the computer.)

Here, for instance, are a few ways to implement a data stream:

- Arguments to a function call, or the return value from a function call
- Data written in a globally accessible location
- Data read from or written to a disk file
- Data passed along a pipeline
- Data read from a serial communications link or network
- A truck full of magnetic tapes
- A (voice) telephone call
- A letter, sent once a month
- A standard form filled out and pasted to the outside of a crate
- A few words shouted over a partition

Similarly, you can implement a data transform in many ways:

- A main routine
- A subroutine
- A program in a pipeline
- A separate microprocessor connected via a data link
- A cam
- A human being
- A trained seal

To stretch your imagination a bit, try drawing a hydrocarbon flow diagram of a 1970 Volkswagen Beetle and identifying all of the different kinds of logic used to turn gasoline into kinetic energy and exhaust. Some of the transforms are mechanical, some fluid logic, some electrical, and some chemical. Who knows, you may think of a more interesting implementation for your IBM 3270.

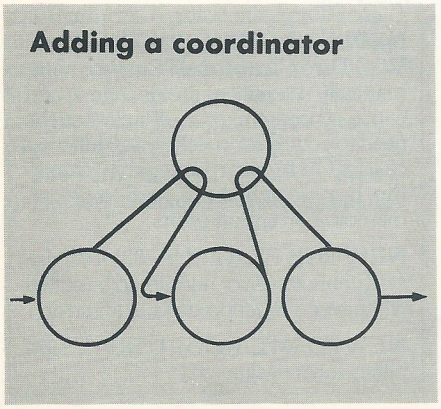


Figure 5.

Back to computers. Even if you confine your attention to data flow diagrams that will be implemented entirely as single programs, you have much more freedom than you may realize. The data flow approach to designing programs tells you to allocate one function for each transform in the data flow diagram. It also tells you to communicate data as much as possible via arguments to the functions and values returned by calling functions. Most important of all, it tells you that the backbone of your program is its data flow. The hierarchy of functions you contrive to implement your program must have the same connectedness as the original data flow diagram.

This raises an interesting issue. You start with a bunch of circles connected by arrows. These circles sprawl across a two-dimensional sheet of paper. (More complex systems may require arrows to cross over each other, or layered data flow diagrams, so even two dimensions may not be enough to represent the data flow.) Your job is to map this network onto a tree of functions. A tree has only one root, the main routine that controls everything. To do the job you must either introduce yet another module, whose sole purpose is to coordinate all of the data transforms as subfunctions, or exalt one of the transforms above all of the others. It would be nice to have some guidelines in choosing among the various alternatives.

Introducing a coordinator at the top is generally the least desirable approach. What you are doing in effect is writing an additional data transform on your data flow diagram, then looping all of the arrows through that single transform, as shown in Figure 5. You then appoint the new transform as the top function in the hierarchy. That violates all sorts of design principles, not the least of which is information hiding.

On the other hand, this is exactly what the UNIX operating system does for you when you write a pipeline. The shell parses the command line description of the pipeline, sets up all of the data streams, and starts each data transform running as a separate program. All data streams loop through the UNIX resident, which calls upon each of the data transforms to resume its activities when it can make additional progress. This design approach is successful, in many cases, because UNIX doesn't have to know any of the detailed internal structure passing through the pipelines it is administering. Each looks like a simple stream of characters being administered by a standard central service.

If this implementation of data streams has acceptable performance, and if the connectedness of the data flow diagram can be modeled by the simple bits of plumbing provided by UNIX, then you can arrive at a quick solution. Many a program has been lashed together in this fashion long enough to tune its specification and prove its worth, before investing

IQCLISP

More Common Lisp features in less space for less money than any other IBM-PC lisp.

- MSDOS portable
- Bignums, 8087 support
- Multidimensional arrays
- Full Common Lisp package system
- Full set of control primitives.
- Keyword parameters, macros
- Save/restore full environments for speed
- STEP, TRACE, BREAK, DEBUG, ADVISE, APROPOS
- Roll-out frees space for invoking MSDOS commands

IQCLISP PACKAGE $300.

LISP *∫q Integral Quality*
P.O. Box 31970
Seattle, Washington 98103
(206) 527-2918

IQLISP

Now with a compiler.

- Compiler comes with source
- Compiler cuts execution time 50-75%, program size 60-80%
- Multidimensional arrays
- Floating point, bignums, 8087 support
- Macros
- Color graphics
- Multiple display windows
- Assembly language interface
- Available for IBM PC or TI-PRO

IQLISP PACKAGE $270.
INCLUDES COMPILER

- **VISA and Mastercard accepted**
- **Generous update policy**
- **Attractive educational license**

CIRCLE 13 ON READER SERVICE CARD

UNIX TOOLS FOR YOUR PC

PC/VI™

UNIX's VI Editor Now Available For Your PC!

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI**—a COMPLETE implementation of UNIX* VI version 3.9 (as provided with System V Release 2).

PC/VI is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI:**

- Global search or search and replace using regular expressions
- Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- Editing of files larger than available memory
- Shell escapes to DOS
- Copying and moving text
- Macros and Word abbreviations
- Auto-indent and Showmatch
- MUCH, MUCH MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI!", "The documentation is so good I have already learned things about VI that I never knew before." — *IEEE Software*, September 1986.

PC/VI is available for IBM-PC's and generic MS-DOS† systems for only $149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

PC/TOOLS™

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

• BANNER	• DIFFH	• PASTE	• SPLIT
• BFS	• DIFF3	• PR	• STRINGS
• CAL	• GREP	• RM	• TAIL
• CHMOD	• HEAD	• SED	• TR
• CUT	• MAKE	• SEE	• TOUCH
• DIFF	• OD	• SORT	• WC

All of these for only $49.00; naturally, extensive documentation is included!

PC/SPELL™

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only $49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** *now* and get **PC/SPELL** for only $1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 5¼", 3½" and 8" disk formats. For more information call today!

*UNIX is a trademark of AT&T. †MS-DOS is a trademark of Microsoft.

CUSTOM SOFTWARE SYSTEMS
P.O. BOX 678 • NATICK, MA 01760
617•653•2555

CSS

UNIX TOOLS FOR YOUR PC

CIRCLE 14 ON READER SERVICE CARD

in more expensive C code. Many a production system in use today remains implemented in terms of UNIX shell scripts that build pipelines, because the solution is maintainable and has performance that is quite good enough for that job.

But let us back up once again and confine our attention to the construction of a single computer program from a data flow diagram. You need to build a tree of modules, which means you need to pick one of them and place it at the top of the tree. You would like to have some guidance in picking the module that goes at the top. You would also like to have some guidance in deciding how to hang the remaining modules off the one at the top. You need, in short, a design method.

The design method that addresses this problem is not outside-in design. What outside-in design does for you is show you how to break up complex modules into simpler modules so that each module is easier to implement. It also leads to modules that better specialize in the important business of information hiding. Given a data flow diagram, you apply outside-in design repeatedly until all of your modules can be structured by left-to-right or right-to-left design. How to organize all of the (now more numerous) modules into a function call hierarchy is the topic of next month's column on top-down design.

To get some sense of the impact of hierarchical organization on the implementation of each module, you might meditate upon the following simple problem. Let us say you need to implement the function that performs the right-most data transform in Figure 2. It takes as input a repetition of atoms and produces groups of atoms as output. If you need to be more concrete, consider the earlier problem of consuming words of text and producing justified lines. Try writing the control structure for this module as:

- An input module you call to get justified lines of text
- A control module that takes words from a subfunction and delivers justified lines to another subfunction
- An output module you call with single words that calls a subfunction whenever it completes a justified line

For extra credit, you can try the same exercise with the cluster of data transforms shown in Figure 4.

To give credit where it is due, I should point out that what I call outside-in design was pioneered by J.D. Warnier. More recently, Michael Jackson and Ken Orr have separately written quite a bit on various aspects of this approach to designing computer programs. The approach known as the Jackson design methodology is largely outside-in design. Those sketchy little data structures I write on the data flow diagrams are handled in much greater detail as Warnier-Orr diagrams. Data flow analysis has seen many contributions from Chris Gane, Trish Sarson, and Ed Yourdon, among others.

Indeed, this column comes closest, of all that I have written so far, to the mainstream of activity in programming design methodology. Next month marks the first anniversary of Programming on Purpose. I will commemorate that anniversary by jumping right into the middle of the mainstream—with both feet.

Never Miss A Compile Again!

BSW-Make, our retargetable *make* utility, speeds software development by automating the chore of rebuilding complex software products after an editing session. No more missed compiles! No more wholesale "just in case" recompilations of the whole product! BSW-Make insures that the minimum set of compilations, assemblies, and links required to correctly update your software are performed after each edit. A major timesaver!

- **Syntax compatible with UNIX *make***
- **Works with any compiler, assembler, or linker**
- **Macro facility for parameterized builds**
- **Indirect command file generation facility overcomes operating system command length limitations**
- **MS-DOS/PC-DOS version only $89.95**
- **VAX/VMS version from $299.95**
- **Not copy protected**
- **Unconditional 30-day guarantee — try it at no risk!**

for free product information, call
(617) 367-6846
Ask for Department C2

The Boston Software Works, Inc.
120 Fulton Street, Boston, MA 02109

CIRCLE 15 ON READER SERVICE CARD

Clarify and document your source listing and get an "organization chart" of your program's structure

with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE,® Fortran and Modula-2 programmers.

Now works with **Fortran**

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

— PC Magazine Sept. 16, 1986

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only $125.00.**

800-257-5773 Dept. 46
In California:
800-257-5774 Dept. 46

MasterCard, VISA, American Express, COD. Add $5 for shipping/handling.

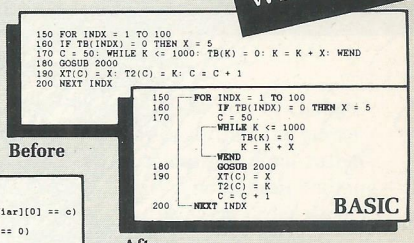
or see your local dealer!

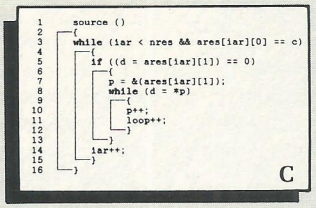
Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

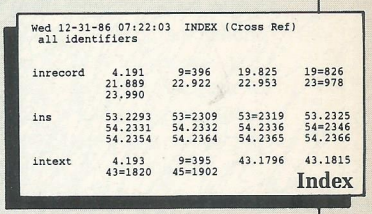
Source Print™

organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:

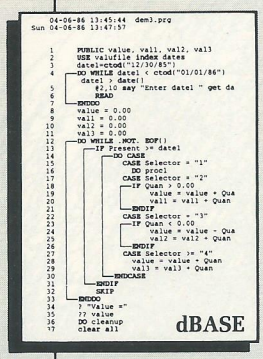
The Index (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called.







$75.00



Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.

Structure Outlining solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

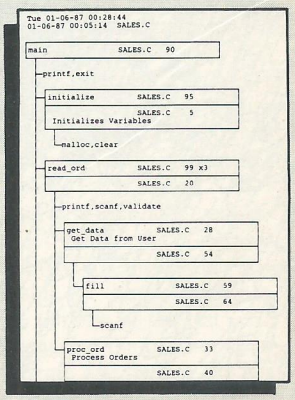
Automatic Indentation of source code and listings reduces your editing time and ensures indentation accuracy.

Plus . . . Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.

Tree Diagrammer™

shows your program's overall organization at a glance. Ordinary program listings merely display functions, procedures, and subroutines sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.

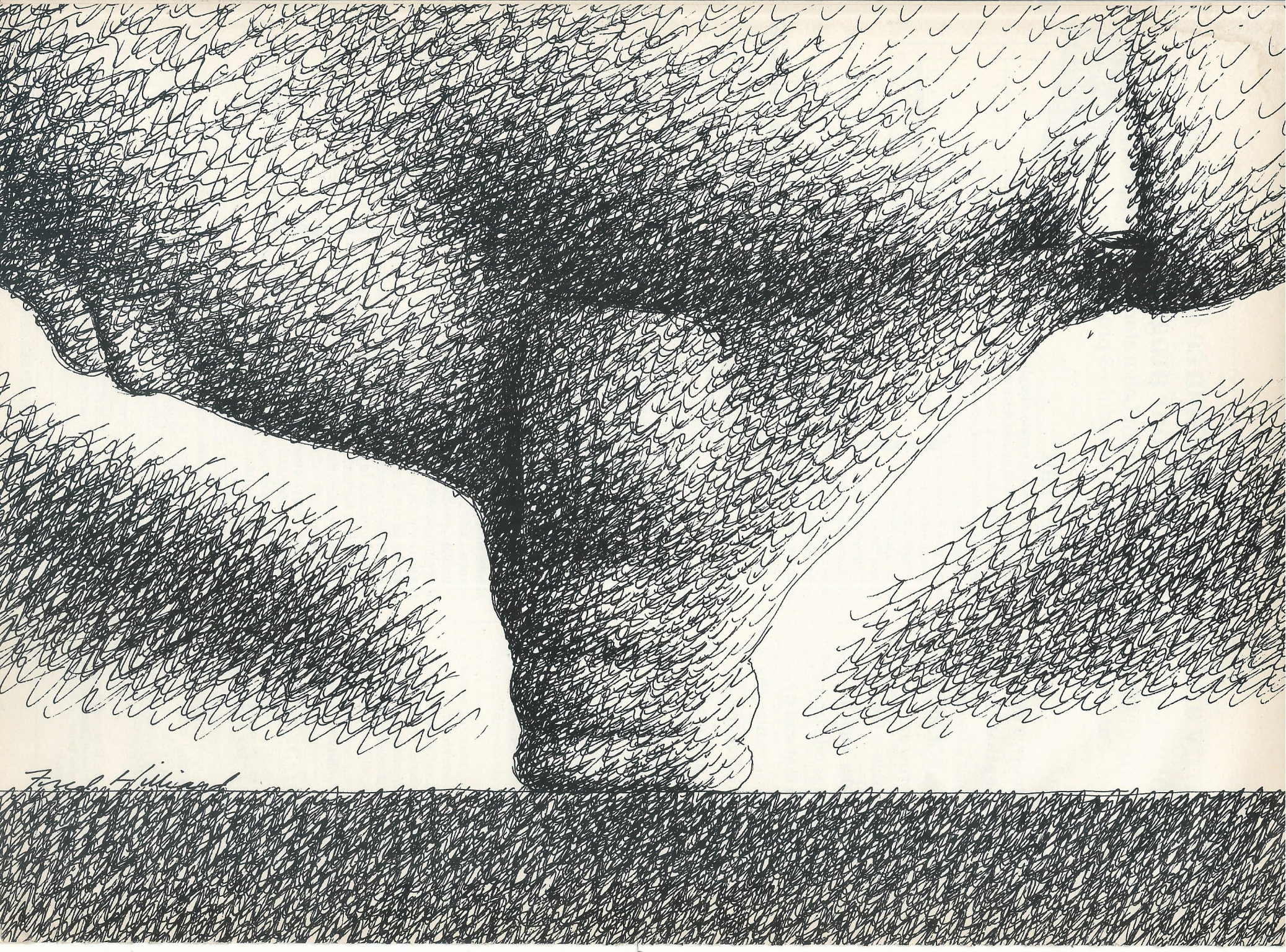
Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.



$55.00

Aldebaran Laboratories 3339 Vincent Rd. Pleasant Hill, CA 94523 415-930-8966

YES! Rush me ☐ Source Print @ $75. _____ ☐ Tree Diagrammer @ $55. _____
☐ Both $125. Ship/Handling $5. For CA add 6% tax _____ Total _____
Name _____
Company _____
Address _____
City _____ State _____ Zip _____
☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express
Card # _____ Exp. Date _____
Signature _____ Phone # _____ 46



Does your programming language present you with annoying "dead ends?"

It may be time to discover the power of Revelation.

When sophisticated programmers first tried database programming languages they said "Too wimpy!" Now they're discovering Revelation's R/BASIC. Yes! It's a Revelation!

Let's start by examining R/BASIC philosophically.

First, it's a structured language, interfacing smoothly with DOS as well as C and Assembler. You'll find commands for structural coding like CASE, LOOP/REPEAT with WHILE/UNTIL, IF/THEN/ELSE and other familiar friends here. And, like all serious programming languages, it has a compiler for fast program execution. A compiler so fast that it can translate 20 lines per second for speedy error detection and correction.

You should expect excellent string handling in a database programming language, but math is another story. Or is it?

Because R/BASIC was developed as a database programming language, naturally you expect it to have powerful string handling capabilities. It does. Revelation's R/BASIC supports both dimensioned arrays and dynamic arrays. Individual strings can be up to 64K in length. But, what you might not expect in a database programming language is a high level of math support. Revelation has it.

R/BASIC offers integral 8087 support for maximum precision. It automatically accesses the 8087 chip for calculations. And, if you don't have an 8087 chip, R/BASIC incorporates an excellent 8087 emulator. Trig functions? Of course.

Subroutine flexibility.

Revelation lets you declare your own internal or external subroutines with argument passing. Subroutines can be developed in C and Assembler, then called directly from an application. And with Revelation's recursive routine calling, you won't have to write as much code.

How much less code?

A long COBOL program typically runs 20,000 lines. A long R/BASIC program typically runs 300 lines. Imagine how much of your development time that kind of difference can save.

R/BASIC will also help you produce a clean program fast. Its completely interactive symbolic debugger traces problems in minutes instead of hours.

Here's another way you can save time with R/BASIC.

By incorporating LOCK and UNLOCK statements in your programs you can change from single to multi-user systems without making program modifications or recompiling.

Is it time for your personal Revelation?

Once you get used to something, it's difficult to change. But if change means progress, it's worth the effort. R/BASIC is a powerful programming language that offers significant built-in database advantages over a "pure" programming language. These advantages can save time during program development and time when the application runs. For the serious programmer, time is money. It's worth your time to take a look at Revelation.

The basics of R/BASIC.

- Compiler.
- Internal and external subroutine calling with argument passing.
- Recursive routine calling.
- Integral 8087 support for maximum precision.
- Support for structured programming.
- No data typing.
- English variable names.
- User-defined functions.
- Full screen editor.
- Support for ASM and C routines.
- Sophisticated string handling includes dynamic array support with a 64K limit.
- Trig functions.
- Alternate syntaxes.
- Direct DOS interface.
- Completely interactive debugger.

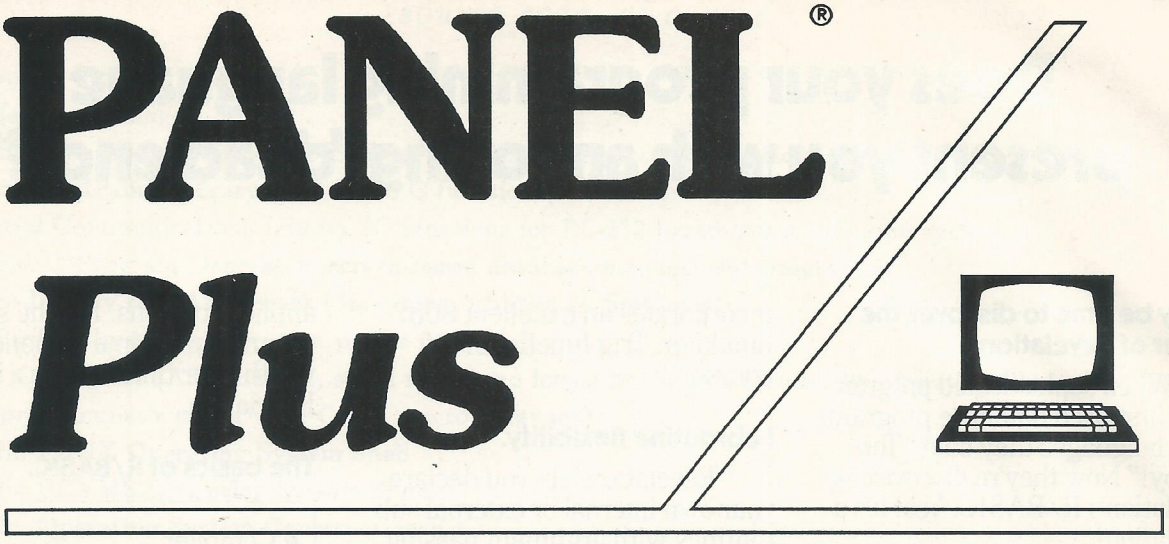
If you would like to sample the power of Revelation, please send $24.95 for our comprehensive Demo/Tutorial. A phone call gets you complete information.

COSMOS™

Cosmos, Inc.
3633 136th Place S.E.
Bellevue, WA 98006, USA
Phone (206) 643-9898
Telex: 185210 (COSMOS MUT)
FAX: (206) 643-7609

CIRCLE 17 ON READER SERVICE CARD

PANEL® *Plus*



Advanced Screen Manager

PANEL *Plus* is a new screen management package which is designed to:

- speed your application development by allowing interactive design of data entry screens, with optional generation of C code to define screen fields and control information.
- make it easy to write high-quality user interfaces in your programs, with pop-up windows and help boxes, scrolling fields, selection of items with mouse button, and extensive editing and user-modifiable validation functions.
- allow your applications written in C to be ported easily to different environments.

PANEL *Plus* is an enhanced version of Roundhill's widely used PANEL screen package. It includes full library source code, and more powerful versions of the interactive screen field editor and C code generator. The function library has been extensively reworked, but programs written with PANEL can be run with minimum change using the new library. All PANEL screen layouts are upwards compatible.

The provision of full library source code, including all variant files for currently supported systems, allows applications using PANEL *Plus* to be moved easily to other C environments. No royalties are payable for the use of PANEL *Plus* libraries when linked into user applications.

PANEL *Plus* will operate in graphics mode using a supported graphics interface, for example to pop-up an entry window over a graphics image. The first release includes interfaces for MetaWINDOW, Essential Graphics, HALO, and Amiga Intuition. Registered PANEL Plus users can arrange to receive updates as new interfaces and new ports are released.

Modular low-level I/O library functions allow independent selection of screen and keyboard support. You can choose from PANEL-compatible, portable I/O, specific system support (eg IBM-PC memory-mapped, Amiga), Unix System V *termio*, or graphics screen output. Modules are also provided for entry and editing of 16-bit Asian characters.

PANEL *Plus* field and screen library functions are smaller, faster, and much easier to use than their PANEL equivalents, and improved library granularity results in smaller programs. There are many new functions, including 'background' processing while waiting for keyboard input, mouse operations, scrolling groups of fields, and menus (with horizontal and/or vertical highlighted bars and help text for each item). There is completely new, expanded documentation.

The price of PANEL *Plus* for MS-DOS, with full library source, is $495. Versions are available for most major C compilers. Please call for prices of PANEL *Plus* for Xenix and Unix systems, and for VAX/VMS. Existing registered users of PANEL will receive a credit against the PANEL *Plus* license fee.

Roundhill Computer Systems Limited
PO Box 8107, Englewood NJ 07631
(201) 569 2265

Roundhill Computer Systems Limited
PO Box 14 Marlborough SN8 1LR England
(0672) 54675

Telex (UK): 444453 AWARE G
Fax (UK): (0672) 54436
BIX: *join roundhill*

Roundhill Computer Systems

CIRCLE 18 ON READER SERVICE CARD

Charles Duff, ACTOR's Shakespeare

By Judy Getts

Chuck Duff is the kind of man you'd expect to run into early in the morning in a university coffee shop, styrofoam cup in hand, talking Hegel with a student.

You'd expect to see him in an art gallery rambling between the neoclassicists and linear chromatists, keen-eyed and at home, or even face-to-face in a folk-jazz cafe if you motioned the waitress aside and asked who is the James Beard behind the watercress crepes.

Duff is not a man you'd expect to be a programming languages developer.

"Most of my background doesn't relate to computers," he admits matter-of-factly, without a trace of apology.

His interest in microcomputers was fired initially by the possibility of using them to play battlefield simulation games. That was in 1977.

Prior to that Duff had been a student of comparative religions, an aspiring Ph.D. in philosophy who was talked out of graduate school by his college advisor, who'd told him he'd be hacking (in the Checker Cab sense) in four years. Duff had also worked as a French chef at some of the toniest eateries in Chicago.

The route to owning his first microcomputer—a 6800-based machine he built from a kit—and to the programming languages he's developed for micros is equally improbable. The one common thread in Duff's journey is his allegiance to object-oriented programming. Nowhere is that loyalty more clear than with his new language for the IBM PC family, ACTOR.

"We had seen that there was a tremendous interest in the object-oriented approach if it could be done efficiently and in a way that the average programmer could understand," he says.

"One of the difficulties with Smalltalk is that it turned into something too big and too complicated by being isolated (at Xerox) for so long. Our interest was in bringing this technology to the mainstream computing world, by making it both more efficient and more accessible."

The result was ACTOR, an innovative interactive language with a Pascal-like syntax and a Microsoft Windows interface. It was designed to be more straightforward to learn than its Smalltalk forebears; cleaner, more compact, and even, if you believe the ads, more fun to use. (For a review of this language, see "ACTOR Goes On Stage" in the March 1987 issue of *AI EXPERT*, pp. 40-44.)

"In ACTOR we've got a minimal set of control structures that anyone can learn in an hour," says Duff. "I would say that for an object-oriented language, its learning curve is very, very low."

Duff and his company, The Whitewater Group, took a year-and-a-half to create ACTOR. One of the biggest hurdles, he admits, was trying to give it the feel of a conventional programming language without compromising its pure object-oriented approach.

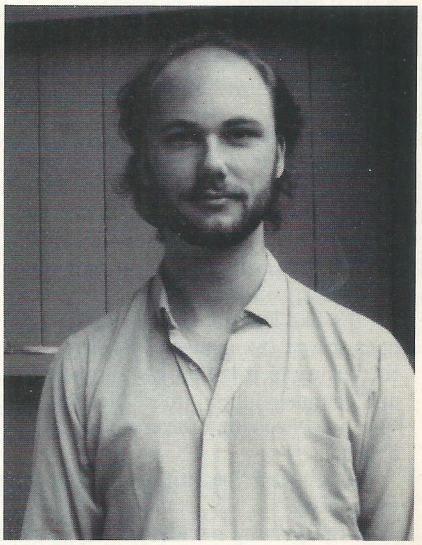
"Other languages like C++ introduce objects into the language, but they've already got this syntax that they're starting with and they have to add objects onto this," he says.

The result is twisted syntax. ACTOR strives for code that "people can look at and say immediately, 'Hey, I know what's going on here.'."

The Whitewater Group is headquartered on a quiet, tree-lined street just off the Northwestern University campus in Evanston, Ill. The company is housed in a former university maintenance building whose tumbledown facade contrasts starkly with its sleekly renovated interior. The building is part of the university's "incubator" project that gives shelter and funding to high-tech start-ups, and The Whitewater Group is one of its beneficiaries.

In his office, Duff talks about the things that led to the genesis of ACTOR—and The Whitewater Group. He's a sandy-haired, lanky man who can talk for hours in perfect sentences—a by-product, no doubt, of his dialectic-steeped past. The story begins around 1982, while Duff was working for a start-up software house called Kriya Systems.

In previous years he'd worked as a systems analyst with a real estate syndicate and had designed software for bank wire-transfer systems before concluding that



he "was not built for the corporate milieu." His vocation as chef had ended years earlier when he entered the University of Illinois to obtain a masters in Information Engineering, learning PDP-11 assembly under the wing of graphics guru Tom Defanti. It was then that he decided that was where his destiny lay.

At Kriya, Duff was supposed to design a new version of the company's bestselling software package, called Typing Tutor, for the recently introduced IBM PC. He started writing it in BASIC, but after a short time grew frustrated with what he calls the "black boxes (of the language) that you can't adjust, and can't change how they work."

He felt strongly that software development occurs more quickly with an interactive language, but the only one able to fulfill that criterion—and that of extensibility—was Forth.

"The beauty of Forth," says Duff, "is that you start out with Forth, but what you end up with by the time you've written your application is a custom language, that is, the language of your application."

Unfortunately, Forth wasn't without its problems—most notably, its physical stack model which, while manageable when crafting simple applications, becomes onerous the more complex the ap-

plication becomes. Its flagrant lack of conventions also bothered Duff.

"Every Forth shop goes off in a different direction and extends it in their own way," he says. "What you end up with is this extremely parochial evolution of the language. It gets distributed out into thousands of different versions."

He nevertheless worked with Forth for a year-and-a-half, using it to develop several different versions of Typing Tutor—until he stumbled onto Smalltalk.

"It just blew my mind," he says. "This is what I'd been aiming at."

What made Smalltalk so revolutionary was that every piece of data, from character strings to compiler, was considered an object.

Think of objects as little experts. Each has its own set of data, assigned operations, and self-contained domain. Because an object's data is isolated from that of other objects, you can erect elaborate systems without conflict between factions.

"It's like Forth in that everything is open to you (to change)," he says, "but the beauty of it is that it's all given form and channeled into a more elegant structure by conceiving of everything as an object."

He calls it a "beautiful way to manage complexity."

In the ensuing months Duff immersed himself in Smalltalk, wondering, "How can I turn the Forth environment into something a lot nicer by using object-oriented principles?"

The result was Neon, a Smalltalk-Forth hybrid—more Forth than Smalltalk—for the Apple Macintosh. At the time the only development environment for the Macintosh was the Lisa-Pascal route, and Duff found that environment "incredibly unwieldy."

Neon was his answer to writing a version of Typing Tutor for the Mac.

The language's first user—besides Duff—was an astronomer at Yerkes Observatory who used Neon to design a remote control panel for manipulating astronomical instruments. Its window-interface simulated mechanical controls and the program communicated with the instruments via a serial link.

What Neon inherited from Smalltalk was mostly philosophy. It used conventional Forth words augmented with the Smalltalk concept of method. Not everything in the language was treated as an object, though—*classes*, for instance, categorize objects and hold their methods and data maps, and in Smalltalk are themselves considered objects. But while the language relied on the familiar stack model, it was more symbolic than physical, referring to objects on the stack by names rather than position.

Neon may have been one of the most inspired languages to come to micros (indeed, Duff claims it was the first object-oriented language ever sold for a microcomputer), but its fate was not so propitiously charted. A week after its first shipment, Duff left Kriya Systems, as did the company's marketing director, Mark Achler, a former Apple employee who had mapped an ambitious public relations strategy for the language.

Kriya was in the throes of internal disarray. Neither Duff nor Achler had been paid in three months, and when the company was bought out Duff was forced to sue to regain royalties to Neon. In his low-key way he calls the episode "kind of painful."

At that point he and Achler decided to forget Neon and formed the Whitewater Group to develop a wholly new language. The fruit of that venture is ACTOR.

Duff and Achler's design strategies for ACTOR aimed to develop a pure object-oriented language that would be efficient, practical to use, based upon a simple syntactical model, and provide dynamic memory management.

"What I found with Neon," he says, "was that the parts of Forth that occurred along with it were really repugnant to a lot of people. Unfortunately, Forth has been cubbyholed and gotten a bad reputation in a lot of circles. Being associated with Forth can be the kiss of death for a mainstream language."

With ACTOR he set out to keep only the aspects of Forth that he felt were useful, like the threaded interpreter.

Duff learned other lessons from Neon that he applied to ACTOR. One was the need for dynamic memory-management.

"To do any kind of sophisticated work like simulations and artificial intelligence, automatic memory management is really essential," he says. "In Forth, there've been a million different extensions for heap kinds of memory management, but nothing is standardized."

One of the greatest difficulties in building ACTOR, he found, was implementing a concurrent garbage collector.

"You don't see the segmented nature of the 8086 when you're in ACTOR," he says. "ACTOR can support the full address space of the machine, so we had to hide the segmented architecture."

The garbage collector that ACTOR ended up with, according to Duff, is "virtually transparent. Very rarely will you see it pause at all."

Another important lesson he learned from Forth was the reverse-Polish notation.

"We had to get rid of that," he states. "It was just not acceptable. It ties you to a physical stack model."

In hunting for a new syntax, something more conventional than that of Forth and less quirky than Smalltalk's, Duff looked at a variety of languages, from Ada to ALGOL. The most suitable, he decided, would be a distillation of Pascal and C.

In ACTOR, he says, "many things read exactly as they would in C or Pascal. For instance, a *Max* function—'give me the maximum of two numbers.' In ACTOR you say *'max (a,b)'* just like you would in those languages."

He was also determined to keep the language's linguistic tree simple.

"One thing I really believe in strongly in a language is that you do not want to proliferate a lot of features. PL/I is an extreme example. It gets so unmanageable, there are so many interactions between features that after a while you've built a Frankenstein."

ACTOR has just three basic control structures: basic-conditional, a general-purpose loop, and a case-type statement. Says Duff, "I wanted to keep the number of different syntactical elements and things that a programmer had to learn to a minimum." He adds, "There aren't any hidden tricks that will get you if you're not careful."

As for a windowing environment for the language, he found himself with some tough choices. He could either build one into the language, as in Smalltalk, or else hook his trailer to an existing windowing product.

"I was disturbed by the idea of throwing yet another proprietary windowing scheme at the world," he explains.

Deciding that Microsoft's Windows was the windowing environment most likely to catch on as a standard in the IBM PC world, he decided to build the language under that.

Although many object-oriented principles are working in Windows, which in some ways made it easy for Duff logically to plug into, it was a challenge to create an interactive programming environment under the operating system—possibly the trickiest challenge in ACTOR. The reason was simple, says Duff: "They have their model and we have ours."

He nevertheless found the ACTOR-Windows pairing to be one of the most gratifying elements of the project.

"It gives you this interactive laboratory," he says. "It's like a chemistry set. You've got all of these bottles with different kinds of windows so that you can do exciting things very quickly."

Programming with the Windows-ACTOR team, he claims, brings immediate gratification, the kind that doesn't come with traditional languages like C that entail editing, compiling, linking, etc.

"In an object-oriented language you can build up a much higher-level model, so you just send a message—*'show'*—to a window and it takes care of all the details and the interface to specific mecha-

Architecture of Wendin's Concurrent I/O System

Wendin's Operating System Toolbox makes it possible to write a concurrent driver for any I/O device.

Wendin's Operating System Toolbox kernel is composed of three major components: the scheduler, the memory manager, and the I/O subsystem. This month we will show how the I/O subsystem works, at the QIO (Queued Input/Output) device driver level.

The I/O system supports concurrent and waited I/O operations from multiple processes simultaneously. Its architecture was derived from the VAX/VMS QIO system, which was developed by Digital Equipment Corporation with these ideas in mind. The whole QIO mechanism takes advantage of many of the kernel's capabilities, including ASTs (Asynchronous System Traps), and event flags (which are used to signal I/O completion).

Before a program can perform I/O operations on a device, it must assign a channel to the device. A channel is simply a number, similar to an MS-DOS file handle. The operating system uses an assigned channel number as an index into an array of channel control blocks (CCBs) to get information about the channel and the corresponding device.

After assigning a channel with the SYS__ASSIGN system service, the user program can make calls to SYS__QIO or SYS__QIOW (for concurrent or waited I/O) through the INT FF software interrupt. The kernel calls the routine EXEQIO, which validates the caller's parameters and I/O privileges. If the call parameters are valid and the channel number corresponds to an assigned CCB, EXEQIO will deliver a KERNEL mode AST to the routine ASTQIO in the same process context. This allows the driver to run on its own stack, but in the context of the calling process. ASTQIO receives all information about the I/O operation in an IRP (I/O Request Packet), created by EXEQIO from the user program's parameters.

```
void ast_qio (packet)
  struct irp *packet;
{
  struct ccb *c;
  struct ucb *u;
  void (*driver)(struct irp *);

  c = &io_chantbl [packet->chan];
  u = &io_unittbl [c->unit];
  driver = u->driver;
  (*driver)(packet);
} /* ast_qio */
```

Listing 1. How ASTQIO dispatches the device driver given a channel number.

ASTQIO uses the channel number to find the corresponding CCB, and obtain a pointer to a data structure describing the device assigned to, in the format of a unit control block (UCB). The unit number is used as an index into an array of UCBs, much as the channel number is used as an index into an array of CCBs. The UCB contains information about a specific device, including what general type of device is being referenced (for example, a terminal, mailbox, or disk), and a pointer to the device driver code. This double indirection (the channel obtains a CCB, which in turn references a UCB, which finally references the device driver) allows several channels to be assigned to the same device at the same time, and for any number of UCBs to share the same device driver code (for example, the disk driver defines a UCB for each disk drive, but there is only one device driver for all of them).

Once ASTQIO has obtained the UCB for the requested device, it calls the device driver directly. The device driver is responsible for determining what to do based on the I/O function given in the IRP. The device driver will typically call the system BIOS, or interface directly to the hardware, to perform an I/O operation.

Finally, the kernel must signal the user process that the I/O is done. The SYS__QIO services allow two methods for this: an event flag can be set, or an AST can be executed when the operation finishes. The kernel routine STOPIO examines the IRP associated with an operation, and determines which event flag to set, and whether to queue an AST to the user process. Finally, the IRP is removed from the system I/O queues, and returned to the memory pool. Control is then passed back up the chain and returns to the user program.

A device driver can be written to talk to any device, based on standard models provided by the Operating System Toolbox kernel. Examples include mailboxes (functionally equivalent to UNIX pipes), terminals, disk drives, optical disks, or even expanded or extended memory. Basically, the minimum I/O functions that the driver needs to support are the read and write block functions. For some devices (like disks) there will be a distinction between virtual, logical, and physical I/O. For others (like mailboxes), all of these functions will perform the same operation. You can decide how you want a device driver to work, and it's easy to integrate a driver into the toolbox kernel.

Next month, we'll continue with the internals of the toolbox. If you'd like to learn more about operating system design and architecture, pick up a copy of Operating System Toolbox from Wendin and follow along.

Operating System Toolbox: $99
Wendin, Inc.
P.O. Box 3888
Spokane, WA 99220
(509) 624-8088

CIRCLE 19 ON READER SERVICE CARD

nisms in Microsoft Windows. It's really brought Windows alive," he adds.

In addition to the Windows connection, ACTOR also provides run-time windows that allow programmers to forgo the Microsoft product and craft their own windowing applications.

Right now The Whitewater Group is working to bring ACTOR to UNIX and the Macintosh. Even more ambitiously, they're assembling a generic windowing application model that will enable any ACTOR code to conform to that model so the code will run on any machine ACTOR runs on. "That's something that objects can do," says Duff, "because they have this logical vs. physical orientation."

The company is also trading on its ties with Northwestern University and collaborating with researchers there to craft AI tools geared specifically to expert systems. Although they have no products on the market at present and are remaining mum about exactly what's on the drawing board, they expect to release some products in the future.

Another ongoing project is the creation of an ACTOR class library that will be made available for free to anyone who purchases the language.

Asked about the future of object-oriented languages as a whole, Duff, as would be expected, is unequivocally bullish.

"The future's very bright," he states. "People are tired of doing things the way they have been doing them."

He mentions several studies which show that an object-oriented programmer's productivity is several times that of a programmer working with a conventional language.

"There is a learning curve associated with object-oriented languages," he admits, "because you have to learn all these classes and what they can do. The beauty is that once you learn what one class can do, you have a technique that can be applied uniformly throughout the system. Consistency is a strong point. Once you get that under your belt, you can absorb what the system already has in it. You have to invent very little new code to do something new."

Past objections to object-oriented languages on microcomputers have focused on the truckload of memory they demand. Indeed, to use ACTOR and Windows together, you need a system with at least 640K.

But Duff anticipates that as micros grow more powerful, RAM requirements will become less vexing. He also anticipates object-oriented languages becoming de rigueur in the artificial intelligence world.

"I think objects can bring a tremendous amount of sophistication to the AI field, and that's just starting to be appreciated," he says.

While many breeds of LISP invoke some object-oriented principles, Duff considers them, and the language itself, logically confused.

"One of the things I don't like about LISP," he says, "is that there's a very heavy emphasis on changing the environment dynamically, for instance, changing things on the fly. You end up with a system that, after a while, you've no idea how it got to be the way it is.

"When you start looking at applying languages like this in areas such as the military, the handling of hazardous ingredients, or monitoring patients on operating tables, you better damn well have a good idea of what that system's going to do at runtime.

"I think you can do anything in an object-oriented language that you can do in LISP," he continues, "but in a more contained, more rigorously structured manner that lends itself to explanation and examination."

Explanation. Examination. Duff sounds like an Aquinas here, even more so when he says, "At this point I feel like I'm always going to be an object-oriented programmer." He then hastens to add, "I think this approach, this philosophy, is something that's going to be around for a very long time."

DIRECTLY ACCESS dBASE III DATA FILES

db DOS $39.95
CREATE, VIEW AND EDIT dBase III data files from the DOS prompt. Use fast and powerful full screen editing with search, append and direct GOTO capabilities. Output to the screen or printer at your choice. All this from the DOS prompt.

db PASCAL $29.95
TURBO CHARGE YOUR TURBO PASCAL FILE ACCESS by using dBase III data files. Simple dBase III file access allows report output with turbo speed. Includes a create utility which writes record structure code automatically. This allows field access using dBase field names. Sample programs provided and source code is included.

ORDER NOW
Phone orders
1-800-433-6854
In Arizona
(602) 435-2370
Or send check or money order to:
LogicPath • P.O. Box 1267
Chandler, Arizona 85224-1267.
We welcome Visa/MC or COD in certified US funds only. Add $2.50 for shipping per order. In Arizona add 6% sales tax. Call or write for other products or additional information (602) 435-2370.

LOGICPATH

dBase III is a trademark of Ashton-Tate.
Turbo Pascal is a trademark of Borlan Intl.

CIRCLE 20 ON READER SERVICE CARD

MICROSOFT LANGUAGES NEWSLETTER VOL. 2, NO. 6

News about the Microsoft Language Family

New Microsoft® QuickBASIC 3.0 Includes Integrated Debugger and Editor Enhancements

It's hard to improve on a great product like Microsoft QuickBASIC, but we've added and enhanced features in Version 3.0 that make developing programs faster and easier than ever. Integrated debugger enhancements let you pinpoint errors by tracing or animating through your source code. Now you can observe the contents of your program's variables as the program is running. And set dynamic breakpoints at runtime to easily stop the program where you want. You don't have to add STOP statements to your programs and wait for another compilation anymore. With the debugger completely integrated into the compiler, you can start debugging your program while it's running simply by pressing CTRL-BREAK.

Microsoft Quick-BASIC's built-in editor has been enhanced to support overtype as well as the existing insert mode so editing your programs is easier. In contrast to other compilers that give up after finding a single error, Microsoft QuickBASIC keeps track of all errors found during compilation. You can correct all the errors at once instead of recompiling for each error. And the Microsoft QuickBASIC Version 3.0 editor is compatible with SuperKey,® ProKey,™ and SideKick® programs.

As in Version 2.0, Microsoft QuickBASIC 3.0 supports multiple-module programming. Separate compilation lets you divide your program into pieces that are compiled independently. These pieces can be integrated into other BASIC programs without recompilation. The linker in Microsoft QuickBASIC makes it easy to add Microsoft Macro Assembler object files to your BASIC programs. Just assemble your Microsoft Macro Assembler routines and let the linker incorporate them into Microsoft QuickBASIC automatically. There's no need to convert your Microsoft Macro Assembler routines into COM files or write them as hexadecimal numbers in your BASIC source file as in other compilers.

Math Coprocessor Support in Microsoft QuickBASIC 3.0 Speeds Programs

Microsoft QuickBASIC now has support for the 8087 and 80287 math coprocessors. The full 80-bit IEEE math support of the math coprocessor is needed for programs that demand the most precise calculations. In addition to this, you may use the existing Microsoft Binary Math support for fast 64-bit math or the new 8087 emulation routines for more accuracy when you don't have a coprocessor.

There have also been many dramatic speed enhancements in Microsoft QuickBASIC 3.0, especially in graphics and math. For example, Version 3.0 is 118% faster than Version 2.0 for circles. And with 8087/80287 math coprocessor support, the floating-point math is 160% faster than in Version 2.0.

New Microsoft QuickBASIC Language Extensions Make Programs Easier to Read

A number of new control structures such as SELECT CASE, DO WHILE, DO UNTIL, LOOP WHILE, LOOP UNTIL, and EXIT have been added to Microsoft QuickBASIC 3.0. These are statements similar to those found in Microsoft C and Microsoft Pascal. These statements, in addition to the block IF/THEN/ELSE/END IF (which was incorporated in Version 2.0), make structured programming easier.

The new named constants make your programs more readable and easier to maintain. Subprograms and multi-line functions have true local variables and allow you to call them by name and pass them parameters. These C-like structures let you write programs that are more well organized.

For more information on the products and features discussed in the Newsletter,
write to: Microsoft Languages Newsletter
16011 NE 36th Way, Box 97017, Redmond, WA 98073-9717.
Or phone:
(800) 426-9400. In Washington State and Alaska,
call (206) 882-8088. In Canada, call (416) 673-7638

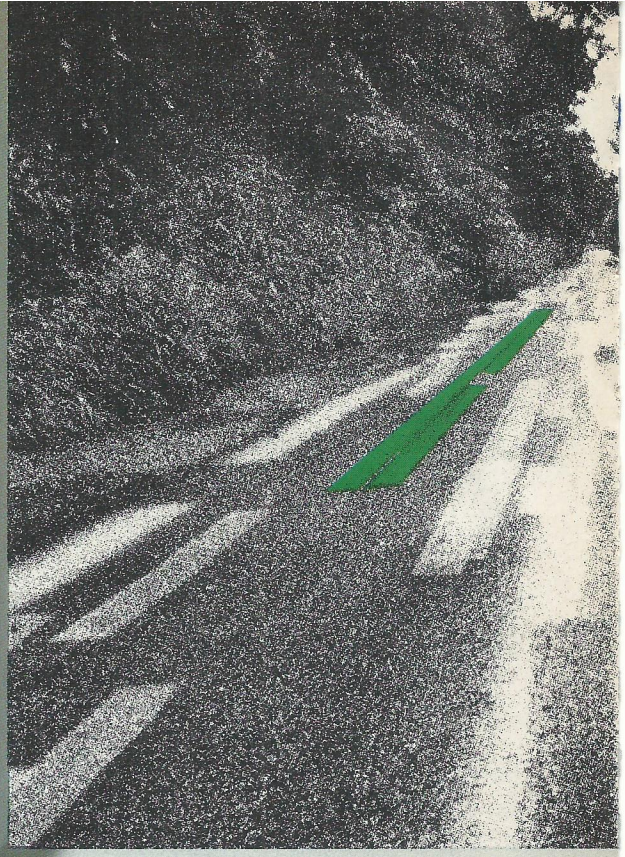
Latest DOS Versions:	
Microsoft C Compiler	4.00
Microsoft COBOL	2.20
Microsoft FORTRAN	4.00
Microsoft Macro Assembler	4.00
Microsoft Pascal	3.32
Microsoft QuickBASIC	3.00

Microsoft and the Microsoft logo are registered trademarks of Microsoft Corporation. ProKey is a trademark of RoseSoft, Inc. SideKick and SuperKey are registered trademarks of Borland International, Inc.

Look for the Microsoft Languages Newsletter every month in this publication.

Microsoft®

SAVE TIME



By Christopher Vergoz-Rekis

Question: Is it possible to run the DOS command system from within a program written in, say, BASIC or Lotus 1-2-3 without having to end that program? The answer is "Yes," using another program called PUSH.

The name PUSH is borrowed from the DEC System-20 command *PUSH*, which pushes the user to a free area of memory on the operating system without destroying the current process or program. This same feature can be made available on the PC using several of the system calls provided through the DOS system calls (interrupt 21h).

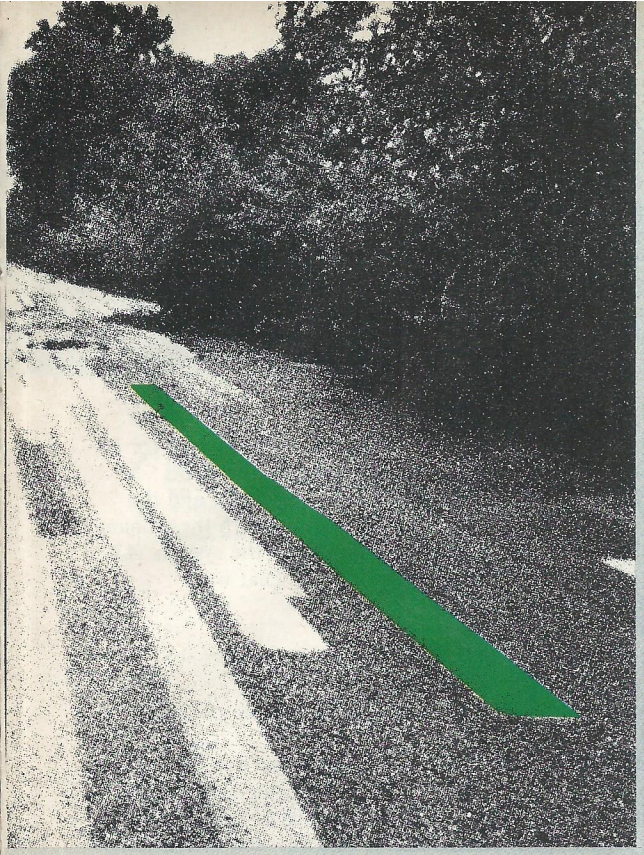
The attraction here is the time saved in not having to reload and execute a program whenever that program is reentered. Consider the time spent loading and reloading a text editor when or unraveling a knotted programming problem.

This article explains how to use the DOS system calls for memory management and process control for loading the DOS command system upon demand, without interfering with the current program. The DOS functions for memory management give the programmer the ability to both reserve and free areas of memory being used by the operating system. Process control is used to redirect requests made to the BIOS interrupt for input from the keyboard and to run another program (for example, COMMAND.COM).

The problems to be resolved in suspending a program and running DOS are partitioning memory, installing an interrupt-handling routine, and loading child processes.

Partitioning memory involves modifying the area of memory used by PUSH.COM, dividing the memory remaining into two portions, and reserving the remaining portions for later use. One of the reserved portions is used to run the first program (for example, BASIC or Lotus 1-2-3). The other portion of reserved memory is set aside for running COMMAND.COM when needed.

Installing an interrupt-handling routine is a matter of replacing the address of the BIOS keyboard interrupt with that



This program pushes the user to a free area of memory in DOS without destroying the current process or program

WITH PUSH

of the user-defined interrupt-handling routine to be installed.

Once installed, the user-defined routine performs two functions. First, all requests to the BIOS interrupt are screened to monitor for the key press indicating the user's call to load the DOS command system. Second, if the key press is given, the necessary steps to run the DOS command system while suspending the current program are then performed.

Running a program (also known as a child process) from within another program is straightforward but requires several pieces of information in predefined locations for processing by DOS. Here, the first child process loaded is the program the user wishes to suspend or retain in memory while loading the DOS system. The second program is called COMMAND.COM.

The steps necessary to implement each of these operations are described in detail, along with the options available to the programmer.

Partitioning memory

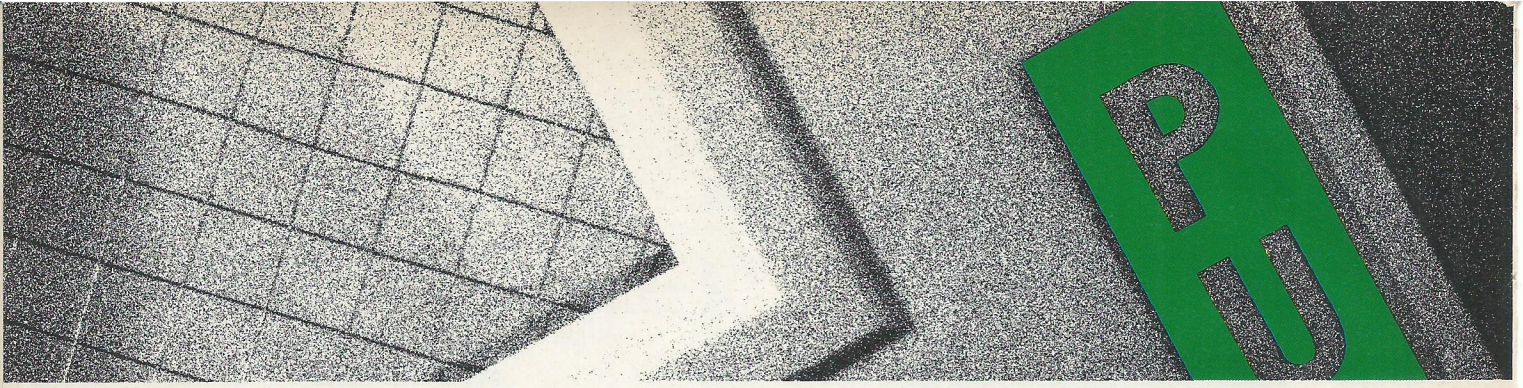
After the name of the program to run is retrieved and stored, memory is modified to free space for two additional programs following PUSH.

Memory is made available through DOS in units designated as memory blocks. At the segment boundary of any one memory block are 16 bytes (a paragraph) known collectively as the memory control block.

The memory control block provides information on the immediate memory block and the location of the next block of memory (if any) available to the operating system. This communication effectively chains the memory blocks through their respective control blocks. If any one control block is destroyed, then the succeeding memory blocks are lost to the operating system. The only remedy when this occurs is to reboot the operating system.

When DOS loads a program into memory, it initially allocates all of the system's available memory to that process. To run another program it is necessary to shrink the size of the memory block used by the current program, which releases memory for later use. Since two child processes are to be created, a portion of this newly released memory must be allocated for later use. If not, when the next child is created it will be given all available memory.

The result of this is no memory space in which to load and execute a second



child, defeating the purpose of the exercise. Reserving a memory block for later use controls the memory space allocated to the first child process created. In other words, reserving memory serves the function of limiting the space given to the first program to be executed, thus setting aside memory for the DOS command system to be loaded at a later time. Functions 4Ah and 49h are the two DOS system calls required to perform each of these operations.

Modifying memory blocks

To change the size of the current program's memory block using DOS system call 4Ah, two pieces of information are required: the location of the beginning of the memory block and the number of 16-byte units to allocate to the newly defined memory block. To obtain the segment address for the start of the memory block, it is necessary to use the address of the program segment prefix (PSP).

The PSP is the first 256 bytes of the process currently loaded in memory. It holds information regarding the program and the DOS environment that is available.

When a program is first loaded, the registers *ES* and *DS* hold the segment address of the PSP. As a result, the registers *ES* and *DS* are pointers to the segment of the memory block to be modified. If these registers are to store different values, then the location of the PSP should be saved in a memory location for this purpose. For the present, it is assumed that *ES* holds the segment address of the PSP and of the memory block to be modified.

Determining the number of 16-byte blocks ("paragraphs") is equally straightforward. One method particularly suited to COM type files is to place a label at the end of the code that is to remain resident when the program's memory block is reduced.

To determine the number of paragraphs needed to house the resident portion of the program code, first divide the offset value of the label at the end of the code section (from the PSP) by 16 (for 16-byte intervals). To adjust for round-off error, the number of paragraphs is incremented by at least one. If the subsequent program accidentally overwrites memory in or near your program code, the results may be unpredictable when the program is reentered.

One other method (suited to EXE files) in determining the number of paragraphs to preserve is to subtract the PSP from another segment in the program. Since segment addresses are essential 16-byte increments, subtracting the PSP segment address from the address of a segment results in the number of paragraphs without division or adjusting for round-off.

In sum, the arguments required to call this function are:

```
AH=4Ah
ES=segment address of memory
      area
BX=requested size for memory,
      in paragraphs
```

Allocating memory

At this point, a portion of the available freed memory must be allocated for later use. In essence, this memory block is a dummy block and serves the function of reducing the memory made available to the next program to be loaded and executed. It is tantamount to having a "Reserved" sign placed at a table in a restaurant until you arrive. DOS requires one argument to execute this system call:

```
BX=size of memory to be allocated,
      in paragraphs
```

The programmer can simply predetermine the memory size to be input to the system call. If no error has occurred, then the carry flag is cleared on return and the register *AX* holds the segment address of the memory block so defined. To return this memory block to the system, it is essential that the segment address be stored in a memory location for later use. Without their respective segment addresses, memory blocks cannot be unallocated when needed.

A second method in allocating memory is to divide the available memory into portions. If the amount of available memory is not known at the time of this call, then generating an error will provide this information. When an error occurs in the execution of this DOS function, the carry flag is set and the register *BX* is returned with the amount of available memory in the system. *AX* holds the error code. That is:

```
(flag set)
AX = (7=arena trashed; 8=not
      enough memory)
BX = amount of available memory
```

If the error generated was not arena trashed (meaning memory control blocks have been overwritten or lost), then a second call to this function will allocate the requested number of paragraphs.

The reader should note that when an arena trashed error has occurred it may be the result of a call to one of the other memory management functions (49h and 4Ah) that may have resulted in a control block being destroyed. The error may not be the result of the call to function 48h; the programmer is advised to look elsewhere for the solution.

In this exercise, half of the available memory will be set aside (allocated). If the second call to allocate memory returns without the carry flag set (no error), *AX* will hold the segment address of the memory block allocated. That is:

```
(Carry not set)
AX = segment address of allocated
      memory block
```

This value is stored in PUSH for loading COMMAND.COM at a later time.

Installing interrupt routines

Prior to loading and executing the first program, some method is needed to communicate the user's intention of running COMMAND.COM. One solution is to install an interrupt-handling routine that screens the requests to the keyboard.

Interrupts are sets of routines that perform many of the operations needed to use the devices available to the computer. Interrupts are analogous to subroutines

or functions utilized in higher-level languages. For example, input and output to the keyboard can be performed through interrupt 10h.

A request to an interrupt-handling routine functions very much like a call to an intersegment procedure, except that the flags register is *PUSH*ed onto the stack before the *CS* and *IP* registers, and arguments are passed through the registers.

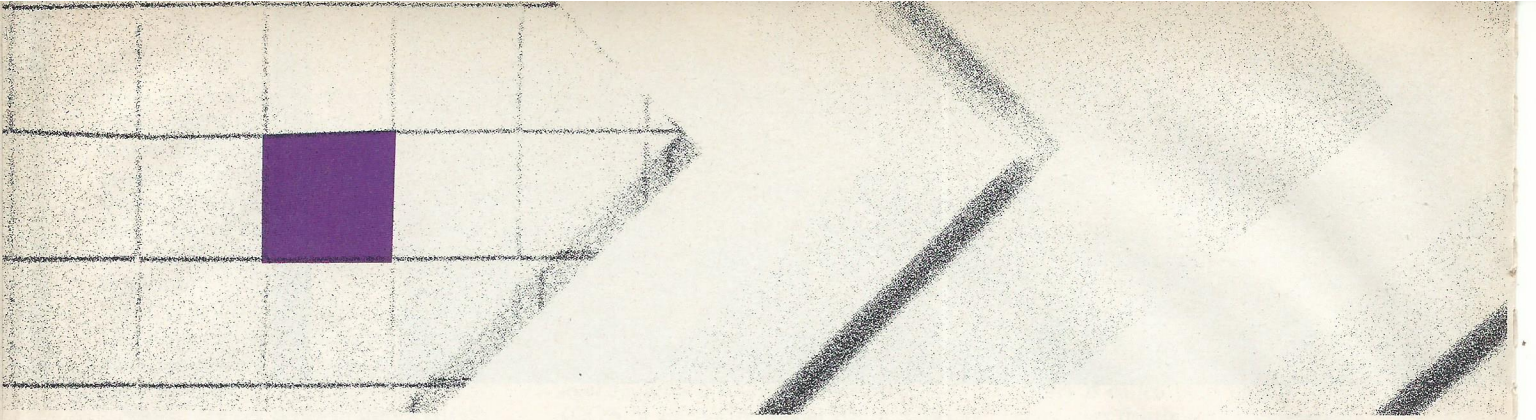
The first instruction for each interrupt-handling routine in memory starts at memory address 0000:0000 in RAM. Each address is stored in a double word as a segment:offset pair. The operating system is designed to process a request to an interrupt by locating its starting address (segment:offset) from the interrupt vector (table). There are no restrictions on what addresses can be stored in the interrupt vector. In other words, the interrupt vector can be changed to suit the programmer's needs.

The interrupt routine described here performs two operations. First, all requests from the child process to the keyboard interrupt routine are redirected to the routine stored in the resident code of the parent process. The first task of this routine is to assess whether the request to the keyboard service routine is for input from the keyboard buffer. If not, continue to the BIOS interrupt routine by means of a far jump. Since the stack is holding the first program's *CS* and *IP*, a far jump will preserve the stack as is. In this way, the interrupt routine will return directly to the first program making the request.

If the call to the keyboard interrupt is for input from the keyboard buffer, imitate a *CALL* to the interrupt by first *PUSH*ing the flag register onto the stack. Then execute a *CALL* statement to the double word address stored in the parent process that points to the keyboard routine. A *CALL* statement places the address of the next instruction in the user-defined interrupt routine and thereby directs the BIOS interrupt routine to return to the current interrupt routine when the request has been processed.

```
PAGE 60,132
TITLE PUSH.com - Loads & Executes COMMAND.COM from within a program.

;        FILE:      PUSH.ASM
;      AUTHOR:      C. VERGOZ-REKIS, PH.D.
;        DATE:      Jul 86 (Copyrighted).  This material is made available to
;                   COMPUTER LANGUAGE for instructional and demonstration
;                   purposes only.  This code is made available as public
;                   domain software for personal/non-profit use.
;     VERSION:      3.1 (All rights reserved.)
; DESCRIPTION:      Allocates memory for loading (and executing) COMMAND.COM
;                   from within the program running as the first child process
;                   (see below).  Interrupt 21h is revectored to screen for ^P.
; Don't forget to type EXIT to get back into your program.
;==>Interrupt numbers (IBM only)
     DOS_FUNC      EQU 21h
     KYBD_DSR      EQU 16h

CSEG SEGMENT                                  ;COM file format

     ASSUME    CS:CSEG, DS:CSEG, ES:CSEG, SS:CSEG
     ORG       100h

; Compilation sequence:      >MASM PUSH;
;                            >LINK PUSH;    {"No Stack" warning generated!}
;                            >EXE2BIN PUSH PUSH.COM
;                            >DEL PUSH.EXE

PUSH:
     JMP       SETUP                          ;initialize

;=================================[  DATA  ]=================================

     FIRST_CHILD_SS   DW    0          ;first_child registers
     FIRST_CHILD_SP   DW    0          ;
     FIRST_CHILD_DS   DW    0          ;
     FIRST_CHILD_ES   DW    0          ;
     FIRST_CHILD_PRG  DB   65 DUP (0)  ;file to 'load&exec'.
     FIRST_CHILD_ARG  DB    0, 0Dh     ;argument to pass [none]

     SECOND_CHILD_PRG DB 'C:\COMMAND.COM',  0 ;full path/filename of DOS
     SECOND_CHILD_ARG DB 0                    ;exec as shell (NOTE: enter
                                              ; >EXIT<return> at the DOS
                                              ; prompt to exit the shell)

     PUSH_KEY    DB 16  ;=<ctrl>+<p> (DOS returns 1 byte values)
     PUSH_SS     DW  0  ;stack registers to save
     PUSH_SP     DW  0  ; ...
     DOS_FLAGS   DW  0  ;save flag register returned by DOS
     BLOCK_SIZE  DW  0  ;size of memory reserved for COMMAND.com
     BLOCK_ADDR  DW  0  ;segment address of memory reserved

  ;==>DOS interrupt address

       INT21_OFFSET DW 0  ;BIOS address for Int 21h
       INT21_SEG    DW 0  ;

  ;==>Pass this info to 'load&exec' DOS function (4Ah)

       PSP_ENVIRON DW 0  ;0=parent environment.
       ARG_OFFSET  DW 0  ;off addr of argument (above).
       ARG_SEG     DW 0  ;seg addr of argument (above).
       FCB1        DD 0  ;dummy (na).
       FCB2        DD 0  ;  "    "
```

Listing 1. *(Continued on following page)*



```
;=========================[ Stack areas ]=========================

                DW  32 DUP (Ø)          ;LOAD_COM stack
     INT_STACK  LABEL BYTE              ;

                DW   32 DUP (Ø)         ;PUSH stack
     PUSH_STACK LABEL BYTE              ;

;=============================[  CODE  ]=============================

RETURN_FROM_SETUP:
     MOV        AX,4BØØh                ;load and execute
     INT        DOS_FUNC                ;call DOS

;==>Reset registers and interrupts before ending PUSH

     CALL       SET_DS_ES               ;reset data registers
     MOV        PUSH_SS,SS              ;restore stack registers
     MOV        PUSH_SP,SP              ; ...
     CALL       RESET_INT21             ;restore interrupt 21h
     INT        2Øh                     ;end program

;*****************************************************************************
;=================[ Set DS and ES registers to CS ]=================

SET_DS_ES PROC                          ;procedure to set data registers

     PUSH       CS                      ;place CS on stack for data registers
     PUSH       CS                      ;
     POP        DS                      ;
     POP        ES                      ;
     RET                                ;

SET_DS_ES ENDP

;================[ Allocates memory block for COMMAND.com ]================

ALLOCATE_MEMORY_BLOCK PROC              ;procedure to reserve memory block

     MOV        BX,BLOCK_SIZE           ;BX=amount of memory to allocate
     MOV        AH,48h                  ;allocate memory
INT21         PROC FAR                  ;procedure to screen Int 21 requests

     CMP        AH,6                    ;DOS functions 6 thru 8 are for
     JB         JMP_TO_DOS              ; keyboard input
     CMP        AH,8                    ;
     JBE        GET_DOS_CHAR            ;

JMP_TO_DOS:
     JMP        DWORD PTR CS:INT21_OFFSET ;proceed to DOS (do not return)

GET_DOS_CHAR:
     PUSHF                               ;place flags on stack
     CALL       DWORD PTR CS:INT21_OFFSET ;call DOS as a procedure
     PUSHF                               ;save flag register from DOS call
     POP        CS:DOS_FLAGS             ;
     CMP        AL,CS:PUSH_KEY           ;compare input byte
     JNE        EXIT_INT21               ;if not, exit procedure
     CALL       RESET_INT21              ;disable call to PUSH
     CALL       LOAD_DOS                 ;run DOS
     CALL       SET_INT21                ;enable call to PUSH

EXIT_INT21:
     PUSH       CS:DOS_FLAGS            ;restore flags returned by DOS
     POPF                               ;
     RET        2                       ;pop flags on return (preserves
                                        ; flags registers)
INT21         ENDP
```

Listing 1. *(Continued on following page)*

Upon return, the *AX* register will hold the value of the key input from the keyboard buffer.

If the *AX* register does not hold the value of the key designated to initiate the second child process, then return to the first child process. If the *AX* register holds the appropriate value, then process the user's request for the second program.

An interrupt-handling routine is installed by first loading the routine into memory for later use. In this example, the routine is stored in the resident portion of the parent process. Next, the address of the interrupt that the current routine will interface to is stored. This address is needed to perform an intersegment *JUMP* or *CALL* and to restore the initial state of the interrupt vector before exiting the parent process. Finally, place the address of the user-defined interrupt-handling routine in the interrupt vector in ROM.

Input and output to the interrupt vector table is accomplished through the two DOS system calls 35h and 25h. A single argument is required for the DOS system call 35h, the interrupt number. The *AL* register holds this value: *AL* = interrupt number. On return, the registers *ES* and *BX* store the requested interrupt-handling routine's segment and offset, respectively, where *ES* = the segment address of interrupt routine and *BX* = the offset address.

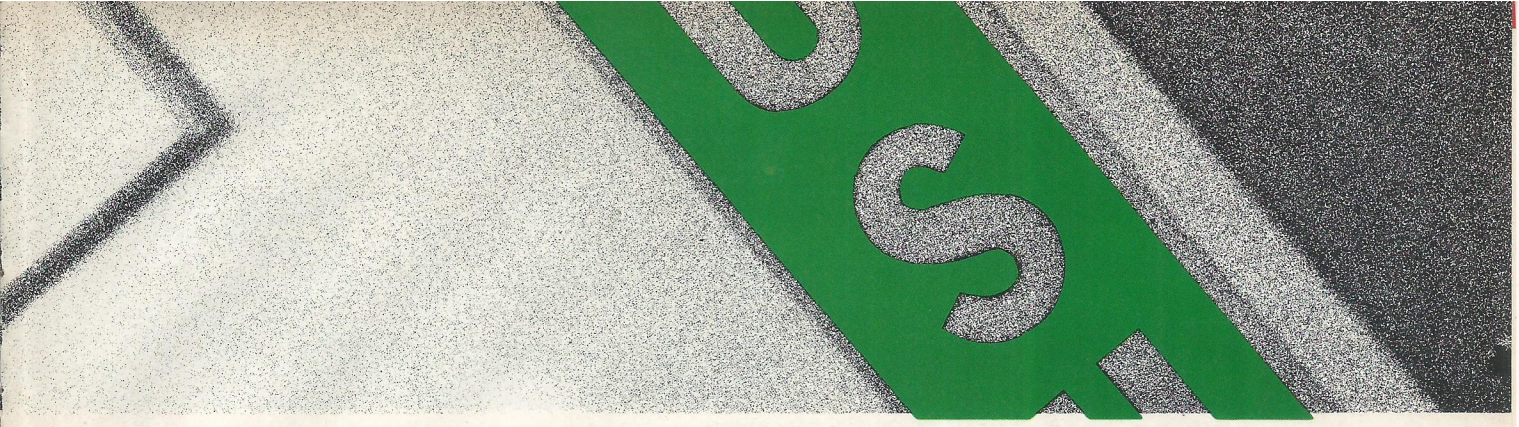
To replace the address of an interrupt-handling routine in the interrupt vector, three arguments are required:

1. The number of the interrupt routine to replace.
2. The segment address of the user-defined interrupt-handling routine (to substitute).
3. The offset address.

This information is passed in the registers *AL, DS,* and *DX,* where *AL* = interrupt number, *DS* = the segment address of interrupt routine, and *DX* = the offset address.

Loading child processes

At this point, memory has been made available for the first program (the first child process) and COMMAND.COM, the second child process. An interrupt routine that screens requests made to the



DOS keyboard system calls has been installed. COMMAND.COM is loaded and executed from the interrupt-handling routine that has been installed to screen requests. The key sequence used here is Control-P. When the interrupt routine identifies this key input, the memory allocated for running COMMAND.COM is unallocated and the second child process loaded in the same manner as the first.

What follows are the steps needed to run the first program, free up memory, run the second program, and allocate memory (again) for later use. The discussion on loading and executing a program applies to the two child processes run from the resident parent process. For the present purposes it is only described once. However, in this program these steps are repeated twice (Listing 1).

The steps to free allocated memory, which are also described, are performed prior to running the second child process. Allocating this same memory for later use (function 48h) is done when the child is exited and control returned to the user-defined interrupt routine.

Loading and executing a program (function 4Bh). To run a program through the DOS system function 4Bh, several pieces of information are required. We need to identify the file name to execute, argument to pass to the program, memory locations of the file control blocks (FCBs) normally contained in the PSP, and segment address of the DOS environment to pass (the information displayed in *SET*).

These last three arguments take a somewhat indirect form in that they are passed to the DOS system function as the address of a block of four arguments that in turn are memory locations to this information.

Certain defaults to simplify this function call are available. The first of these, the name of the program to run, is passed to the DOS function as the segment and offset addresses of an ASCII string (a string of characters ending with byte 0) in the registers *DS* and *DX*, respectively. The ASCII string pointed to by *DS:DX* contains the full path file

```
;=============================[ Load Dos_com ]=============================

LOAD_DOS PROC                           ;procedure to run DOS

     CLI                                ;disable interrupts
     MOV       CS:FIRST_CHILD_SS,SS     ;save current stack registers
     MOV       CS:FIRST_CHILD_SP,SP     ;
     MOV       CS:FIRST_CHILD_DS,DS     ;save current data registers
     MOV       CS:FIRST_CHILD_ES,ES     ;
     CALL      SET_DS_ES                ;reset data registers
     MOV       SP,CS                    ;set stack registers to LOAD_DOS stack
     MOV       SS,SP                    ;
     MOV       SP,OFFSET INT_STACK      ;
     PUSH      AX                       ;save registers from first child process
     PUSH      BX                       ;
     PUSH      CX                       ;
     PUSH      DX                       ;
     PUSH      DI                       ;
     PUSH      SI                       ;
     PUSH      BP                       ;
     STI                                ;enable interrupts

     CALL      FREE_MEMORY_BLOCK                    ;free memory for COMMAND.com
     MOV       ARG_OFFSET,OFFSET SECOND_CHILD_ARG   ;offset of argument
     MOV       ARG_SEG,CS                           ;segment
     MOV       DX,OFFSET SECOND_CHILD_PRG           ;asciz filename to run

     INT       DOS_FUNC                 ;call DOS
     MOV       BLOCK_ADDR,AX            ;AX=segment address of memory
     RET                                ;

ALLOCATE_MEMORY_BLOCK ENDP

FREE_MEMORY_BLOCK PROC                  ;procedure to unreserve memory block

     PUSH      ES                       ;changed by DOS function
     MOV       AX,BLOCK_ADDR            ;segment of allocated memory
     MOV       ES,AX                    ;ES=segment address
     MOV       AH,49h                   ;free allocated memory
     INT       DOS_FUNC                 ;call DOS
     POP       ES                       ;restore ES
     RET                                ;

FREE_MEMORY_BLOCK ENDP

;=========================[ Set interrupt addresses ]=========================

SET_INT21      PROC                     ;procedure to point BIOS to INT21 PROC

     PUSH      AX                       ;registers changed by DOS function
     PUSH      DX                       ;
     PUSH      DS                       ;
     MOV       AX,2521h                 ;set interrupt 21h address
     MOV       DX,OFFSET INT21          ;DX=new address
     PUSH      CS                       ;DS=segment address
     POP       DS                       ;
     INT       DOS_FUNC                 ;call DOS
     POP       DS                       ;restore changed registers
     POP       DX                       ;
     POP       AX                       ;
     RET                                ;

SET_INT21      ENDP

RESET_INT21    PROC                     ;procedure to restore DOS interrupt
```

Listing 1. *(Continued on following page)*

```
        PUSH    AX                      ;registers changed by DOS function
        PUSH    DX                      ;
        PUSH    DS                      ;
        MOV     AX,2521h                ;set interrupt 21h address
        MOV     DX,CS:INT21_OFFSET      ;DX=new address
        PUSH    CS:INT21_SEG            ;DS=segment address
        POP     DS                      ;
        INT     DOS_FUNC                ;call DOS
        POP     DS                      ;restore changed registers
        POP     DX                      ;
        POP     AX                      ;
        RET                             ;

RESET_INT21     ENDP

;=============================[ Interrupt 21h ]=============================

        MOV     BX,OFFSET PSP_ENVIRON           ;address of parameters
        MOV     AX,4B00H                        ;AH=0 (load and execute)
        INT     DOS_FUNC                        ;call DOS
        CALL    SET_DS_ES                       ;restore data registers
        CALL    ALLOCATE_MEMORY_BLOCK           ;save memory for next time

        CLI                             ;disable interrupts
        POP     BP                      ;restore registers
        POP     SI                      ;
        POP     DI                      ;
        POP     DX                      ;
        POP     CX                      ;
        POP     BX                      ;
        POP     AX                      ;
        MOV     SP,FIRST_CHILD_SP       ;restore stack registers
        MOV     SS,FIRST_CHILD_SS       ;
        PUSH    FIRST_CHILD_DS          ;restore data registers
        PUSH    FIRST_CHILD_ES          ;
        POP     ES                      ;
        POP     DS                      ;
        STI                             ;enable interrupts
        RET                             ;return to ...

LOAD_DOS        ENDP

;==>end of resident portion

SETUP:
        MOV     SP,CS                   ;initialize stack
        MOV     SS,SP                   ;
        MOV     SP,OFFSET PUSH_STACK    ;

;==> Command line argument

        MOV     SI,80h                    ;SI=command tail address
        MOV     CL,[SI]                   ;get number of characters
        XOR     CH,CH                     ;CH=0
        ADD     SI,2                      ;DS:SI=first character of argument
        DEC     CX                        ;above CX includes leading <space>
        MOV     DI,OFFSET FIRST_CHILD_PRG ;ES:DI=save filename to ...
        CLD                               ;move bytes in ascending order
        REP     MOVSB                     ;move string of CX bytes

;==> Shrink memory space given to PUSH

        MOV     BX,OFFSET SETUP         ;last byte to save
        MOV     CX,4                    ;divide by 16 by ...
        SHR     BX,CL                   ; shifting bits right 4x's
        INC     BX                      ;add +1 for round-off error
        MOV     AH,4Ah                  ;modify memory
        INT     DOS_FUNC                ;call DOS
```

Listing 1. *(Continued on following page)*

name terminated by a 0 byte. For example:

"C:/PATH/FILENAME.EXT",0

Function 4Bh simplifies the specification of the program's file name by defaulting to the current disk and directory. However, if the full file name (and extension) specified in the ASCII string pointed to is not in the current directory, then no search is performed using the parameters available in the *PATH* command. In response, the function sets the carry flag and returns an error code (for example, 2 - "File not found") in the *AL* register.

The remaining three arguments are input in the form of a seven-word parameter block. The segment and offset address of this parameter block are passed in the registers *ES* and *BX*, respectively.

The first word in the parameter block is the segment address of the environment to pass to the program to be loaded. The DOS environment is simply the output generated by the *SET* command. In memory, the output is stored as consecutive ASCII strings terminated collectively with two 0s. The segment address of the environment is located at offset 2Ch in the PSP. The offset address is always 0 since the DOS environment starts on a segment boundary (offset 0). In this way, DOS can be redirected and a different or modified environment can be created.

An example of this is the redefining of the DOS prompt when the second copy of COMMAND.COM is loaded. To use the environment of the parent process, without alteration, place 0 in this word.

The three remaining arguments in the parameter block are offset and segment pairs pointing to other memory locations. The next two words are the offset and segment addresses of the argument string to pass to the child process. This string is passed to offset 80h in the PSP of the child process. This argument is the same input that follows the file name entered at the DOS prompt, for example:

C:-> EDLIN PUSH.ASM

This string is formatted differently from the strings described so far. The first value is the number of characters following the name of the program (such as COMMAND). The characters that follow are entered literally as they would appear on the command line following the program's file name (such as PUSH.ASM). The last character of the string is 13 (carriage return). The full string pointed to appears in memory as (hex) 08 20 50 55 53 48 2E 41 53 4D 13, where:

```
08          = parameter string length
20 ... 4D   = parameter string
              ("PUSH.ASM")
13          = carriage return
```

(Quotation marks are used to delimit the character string in the example and are not part of the literal string passed to DOS.)

The last four words hold the full address (offset and segment) of two FCBs used by the program. However, the use of FCBs is rare since most current programs use the file handle method of file access. Again, a value of 0 directs the function call to use the FCBs of the parent process (at offsets 5Ch and 6Ch in the PSP).

In sum, the addresses of information to be passed to the PSP of the child process are stored in the parameter block as seven contiguous words, where:

```
;==>Calculate memory for two processes

     MOV       BX,0FFFFh                ;start with largest number
     MOV       AH,48h                   ;allocate memory
     INT       DOS_FUNC                 ;call DOS

     JNC       ALLOCATE_MEMORY          ;if no error, allocate memory

     CMP       AL,8                     ;if error is "Not enough memory"
     JE        ALLOCATE_MEMORY          ;try again (BX=available memory)

     INT       20h                      ;if other error, end program

ALLOCATE_MEMORY:
     MOV       CL,1                     ;divide into two equal parts
     SHR       BX,CL                    ;BX=available memory (paragraphs)
     MOV       BLOCK_SIZE,BX            ;amount of memory to allocate
     CALL      ALLOCATE_MEMORY_BLOCK    ;allocate memory for Command.com

;==>Setup interrupt-handling routines

     PUSH      ES                       ;ES changed by DOS function
     MOV       AX,3521h                 ;get interrupt 21h address
     INT       DOS_FUNC                 ;call DOS
     MOV       INT21_OFFSET,BX          ;BX=offset
     MOV       INT21_SEG,ES             ;ES=segment
     POP       ES                       ;restore ES register

     CALL      SET_INT21                ;set 21h to procedures (below)

;==>Load and execute the first program

     MOV       ARG_OFFSET,OFFSET FIRST_CHILD_ARG  ;address of argument
     MOV       ARG_SEG,CS                         ; to first program
     MOV       DX,OFFSET FIRST_CHILD_PRG          ;DS:DX=seg:off of filename
     MOV       BX,OFFSET PSP_ENVIRON              ;ES:BX=address of parameters
     MOV       PUSH_SS,SS                         ;save stack registers
     MOV       PUSH_SP,SP                         ; ...
     JMP       RETURN_FROM_SETUP                  ;return to resident code

CSEG  ENDS     ;end of segment
      END PUSH ;end of program
```

Listing 1. *(Continued from preceding page)*

Instant Replay™
Version II

Demonstrations
Presentations
Prototypes
Tutorials
Menus
Music

"Indispensable . . . A clear improvement over Dan Bricklin's Demo Program."
PC Magazine

"Incredible . . . We built our entire Comdex Presentation with Instant Replay."
Panasonic

"Instant Replay brings new flexibility to prototypes, tutorials, & their eventual implementation."
Electronic Design

"I highly recommend Instant Replay."
Computer Language

Build tutorial replays of actual programs, or animated prototypes. Memorize and replay keystrokes and pause times. Has the unusual ability to insert prompts, pop-ups, prototypes, user involvement, music, and branching menus into replays.

Includes Screen Maker, Keystroke/Time Editor, Prototyper, Text Editor, Music Maker, Menu Maker, Screen Grabber, Animator, Control and Insertion Guides, Illustrated Manual, Online Tutorial, (Not Copy Protected, No Royalties), 60 Day Satisfaction Money Back Guarantee.

Instant Replay II ™ $149.95
Demo Diskette $5 Dealer Poster $3

Call or Write. We accept Visa, AMEX, Master Card, COD, PO. Dealer Inquiries Welcome.
Outside US pays postage.

Nostradamus
3191 South Valley Street (ste 252)
Salt Lake City, Utah 84109
(801) 487-9662
CIRCLE 21 ON READER SERVICE CARD

MetaWINDOW™
Power Graphics for your IBM PC!

PC TECH JOURNAL
"... a technological tour de force for fast PC graphics."
"Product of the Month"

NO ROYALTIES!

MetaWINDOW is an advanced high performance graphics toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers

High Performance Graphics Capabilities
MetaWINDOW builds on today's state of the art "object oriented" display management concepts, and provides powerful, advanced graphics capabilities similar to those used on the Xerox Star, Apple Macintosh, Sun and Apollo workstations.

- circles & ellipses
- arcs & wedges
- rectangles
- rounded-corner rectangles
- complex multi-polygons.

Power Graphics!
- Display multiple bitmap or "filled-outline" fonts.
- Face fonts for bold, italic, underline or strike-out stylings.
- Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Dynamic mouse/cursor tracking
- Event queue processing
- High speed "bitblt" and "zoomblt" transfers.

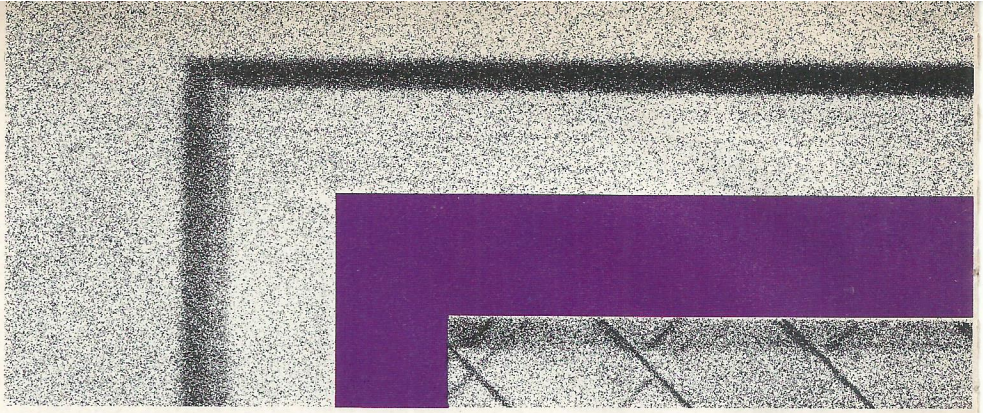
MetaWINDOW comes complete with langauge bindings for 14 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 30 graphics adaptors and input devices.

MetaWINDOW
Advanced Graphics Toolkit
3 disks, 3 260 page manuals - $185

TurboWINDOW
All the features of MetaWINDOW just for Turbo Pascal - $79.50

METAGRAPHICS
SOFTWARE CORPORATION
4575 Scotts Valley Drive
Scotts Valley, CA 95066
408/438-1550

CIRCLE 22 ON READER SERVICE CARD



word 1 = segment of environment
word 2 = offset of command line to be
word 2 = segment of command line
word 4 = offset of default FCB1 (at 5Ch)
word 5 = segment of FCB1
word 6 = offset of default FCB2 (at 6Ch)
word 7 = segment of FCB2

The full set of calling parameters for function 4Bh are:

AX = 4Bh
AL = 0 (load and execute program)
DX = offset of ASCII path name of file to load
DS = segment of ASCII path name
BX = offset of parameter block for file to load
ES = segment of parameter block

Free allocated memory block (function 49h). Function 49h requires only two arguments, the registers *AH* and *ES*, where *AH* = 49h and *ES* = segment address of memory area to be freed. Note that the segment address of the memory block to be freed was output in the *AX* register (function 48h) and stored in a word location in the resident portion of PUSH for this purpose.

Other considerations

Listing 1 was designed as an instructional aid and for that reason may seem terribly inefficient to the serious programmer. The code is meant simply to augment the discussion in this article and clarify any confusion in the text.

The program in Listing 1 does not provide routines to save and restore the screen and cursor position of the first child process. Try using PUSH without saving the screen and see what does (or does not) happen!

Another feature of interest is the ability to screen both DOS and the BIOS keyboard (16h) interrupt routines for ^P. Programs make requests to both DOS and BIOS for the keyboard. The reader should be aware that problems occur if an attempt is made to execute a DOS system call before the current call to DOS (the DOS function making a call to the BIOS interrupt) is completed. In other words, do not jump from the DOS request for keyboard input to the BIOS request (assuming one exists in PUSH) to the *LOAD* DOS procedure. In returning from PUSH's BIOS keyboard interface, the DOS stack has been altered, so the results are unpredictable. Apparently, DOS uses a single internal stack. If system calls are made before the values are retrieved from the previous system call, then those values are overwritten.

This explanation is speculative, but the fact remains that the DOS stack is altered following the procedure to load COMMAND.COM. (A more detailed explanation is beyond the scope of this article. The interested reader is directed to trace the DOS system call and locate the problem.)

If not already apparent, this article provides the reader with the tools to program a windowslike application. For example, consider the preassignment of executable program names to different keys. That is, run dBASE using PUSH and press ^L for Lotus 1-2-3. Or, better yet, display a menu of programs, select two, and press ^Z to change between them.

The programming capabilities described here give the programmer control over memory, interrupts, and processes using DOS system calls. The respective DOS system calls to implement each of these capabilities are relatively easy to use. The program presented here is only one possible application of the power available through DOS.

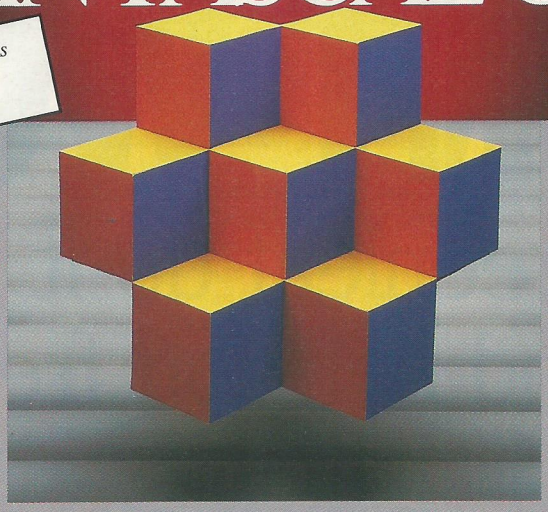
Christopher Vergoz-Rekis holds a Ph.D. in psychology and has research interests in quantitative methods and computer simulation techniques. He is currently involved in PC applications development at RAV Communications, New York, N.Y.

An expanded version of this program is available on the COMPUTER LANGUAGE Bulletin Board Service and CompuServe forum. This version allows the user to save the screen and use both the DOS and BIOS interrupts.

Artwork: Anabella Gonzalez

WHY LOGITECH MODULA-2 IS MORE POWERFUL THAN PASCAL OR C.

"A clear winner... The integrated editor is a joy to use." BYTE Magazine, Jan. '87



APPRENTICE PACKAGE $99

WIZARDS' PACKAGE $199

- **Separate Compilation w/inter-module typechecking**
- **Native Code Generation**
- **Large Memory Model Support**
- **Most Powerful Runtime Debugger**
- **Comprehensive Module Library**
- **Maintainability**
- **Translator from Turbo and ANSI Pascal**

NEW!

APPRENTICE PACKAGE $99

Everything you need to begin producing reliable maintainable Modula-2 code. Includes the Compiler with 8087 support, integrated Editor, Linker, and BCD Module. We're also including FREE our Turbo Pascal to Modula-2 Translator!

NEW!

WIZARDS' PACKAGE $199

This package contains our Plus Compiler—for professional programmers or for those who just want the best. The Plus Compiler with Integrated Editor requires 512K and takes advantage of the larger memory to increase compilation speed by 50%. Our Turbo Pascal to Modula-2 Translator is also included at no extra charge.

NEW!

MAGIC TOOLKIT $99

We've put our most powerful development tools into one amazing Toolkit for use with either the Apprentice or Wizards' packages. Highlighted by our Runtime Debugger, the finest debugging tool available anywhere, the Toolkit also includes our Post Mortem Debugger, Disassembler, Cross Reference utility and Version which keeps track of different versions of one program. Our MAKE Utility figures out module dependencies and automatically selects those affected by code changes to minimize recompilation and relinking. We also provide source code of our major library modules for you to customize—or just play with.

WINDOW PACKAGE $49

Now you can build true windowing into your Modula-2 code. Features virtual screens, color support, overlapping windows and a variety of borders.

ROM PACKAGE AND CROSS RUN TIME DEBUGGER $299

For those who want to produce rommable code. You can even debug code running in ROM from your PC.

Turbo Pascal is a registered trademark of Borland International.

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

To place an order call toll-free:

800-231-7717

In California:

800-552-8885

WIN A FREE TRIP TO Switzerland

HOMELAND OF MODULA-2

Return your Modula-2 Registration Card or a reasonable facsimile* postmarked between March 1, 1987 and May 31, 1987 to be included in a once-only drawing!

Grand Prize: One week excursion for 2 in Zurich, Switzerland including a guided tour of ETH, the University where Modula-2 was created by Niklaus Wirth. European customers may substitute a trip to Silicon Valley, California.

Second and Third Prizes: LOGITECH C7 Mouse or LOGITECH Bus Mouse with Paint & Draw software—a $219 value, absolutely free!

*Write to Logitech, Inc. for a registration card facsimile.

YES! I want the spellbinding power of LOGITECH Modula-2!

☐ Apprentice Package	$99
☐ Wizards' Package	$199
☐ Magic Toolkit	$99
☐ Window Package	$49
☐ ROM Pkg/Cross RTD	$299

Add $6.50 for shipping and handling. Calif. residents add applicable sales tax. Prices valid in U.S. only.

Total Enclosed $_____

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number _____ Expiration Date _____

Signature _____

Name _____

Address _____

City _____ State _____

Zip _____ Phone _____

LOGITECH

LOGITECH, Inc.
805 Veterans Blvd. Redwood City, CA 94063
Tel: 415-365-9852

In Europe:
LOGITECH SA, Switzerland
Tel: 41-21-879656 • Telex 458 217 Tech Ch

In Italy:
Tel: 39-2-215-5622

CIRCLE 23 ON READER SERVICE CARD

QUIT DOING GRUNT WORK.

Let Greenleaf do it for you and set you free.

C Program developers, stop slaving!

Greenleaf libraries have the functions you need — already perfected and in use by winning program developers in major corporations such as IBM, EDS and GM.

Between our Greenleaf Functions and Greenleaf Comm Library, we have over 340 functions on the shelf. Each one can save you time and effort. Money, too.

Many C programmers have told us that, even if they only use one or two functions, our products easily pay for themselves:

The Greenleaf Functions

The most complete and mature C language function library for the IBM PC, XT, AT and close compatibles. Our version 3.0 includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, new disk status and Ctrl-Break control functions plus many more!

The Greenleaf Comm Library

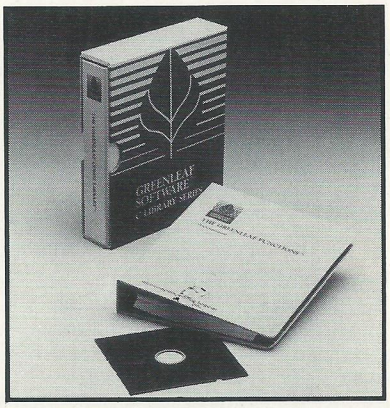
Our 2.0 version is the hottest communications facility of its kind. Over 120 all new functions — ring buffered, interrupt-driven asynchronous communications.

Call Toll Free

1-800-523-9830

In Texas and Alaska, call

214-446-8641



GREENLEAF
Software

Greenleaf Software, Inc.
1411 LeMay Drive Suite 101
Carrollton, TX 75007

If you need more than 2 ports, only Greenleaf gives you the total solution — boards, software, and complete instructions that enable you to build a 16-port communication system.

And no matter how many ports you have, it's virtually impossible to lose information with multiple file transfers. XMODEM, XON/XOFF and Hayes modem controls are featured.

We support all popular C compilers for MS DOS: Lattice, Microsoft, Computer Innovations, Wizard, Aztec, DeSmet and Mark Williams.

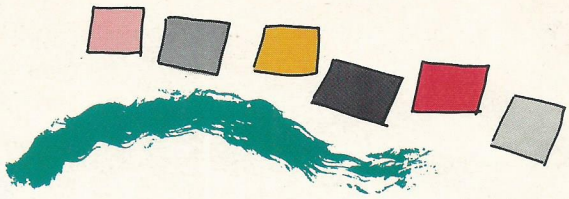
Order today!

Order a Greenleaf C library now. See your dealer or call 1-800-523-9830. Specify compiler when ordering. Add $8 for UPS second day air, or $5 for ground. Texas residents, add sales tax. Mastercard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf	
Comm Library v2.0	$185
Greenleaf Functions v3.0	$185
Digiboard Comm/4-II	$315
Digiboard Comm/8-II	$515

We also sell compilers, books and combination packages.

CIRCLE 24 ON READER SERVICE CARD



Parsing an Equation Without Recursion

By Scott Magruder

Recently, I needed to write a mathematical equation parser for inclusion in a computer tutorial language I was implementing on the IBM PC.

I put this project off for as long as I could because I had always considered it to be one that would take an enormous amount of time and effort—more than it was really worth since all I needed was to be able to set up counters for finding missed questions, calculating grades, and doing a few simple equations.

I began by looking around for some previously published parsers to use as guides. I found one listing of a parser in *BYTE* (August 1985) and another in H. Schildt's *Advanced Turbo Pascal Programming and Techniques.*[1] (Though these are not the only parsers that have been published, they are the only ones I found.)

But both of these routines use recursion, and though the language I had to write the parser in (Turbo Pascal) supports recursion, recursive procedures can be difficult to understand and debug. Further, not all languages allow recursive procedures, so conversion to other languages would not be possible.

My search then led me to books on compiler writing—specifically, *The Anatomy of a Compiler,*[2] 2nd ed., by John A.N. Lee,[2] which provides an excellent description of general parsing techniques and how to implement them. The process shown in this book is straightforward, easy to understand, and efficient.

The program in Listing 1 implements this procedure. The parser allows previously defined variables and advanced functions to be used in the equation. Currently, 100 variables are allowed. Variable names, which are converted to uppercase when found, are limited to 15 characters.

Theory of the parser

(The following discussion is based on pp. 325-337 of *The Anatomy of a Compiler.*)

The parser in Listing 1 requires that numerical values be assigned to each operator. The following values will be used (VOP stands for value of the operator):

Operator	VOP
> <	0
=	1
+ −	2
* /	3
~	200
^	5
(	+10
)	−10
EXP and other special functions	100

In the analysis table, the operator > is used to implicitly mark the beginning of the equation, < to mark the end of the equation, ~ to represent unary minus for convenience, and ^ to raise a value to a power. *EXP* is the exponential function. Other functions, such as *SINE, COSINE,* etc., can be included easily.

These operator values allow correct ordering of the different operations. For example, the equation $Y = 3 + 5 * 9$ requires that 5 be multiplied by 9 first. This value is then added to 3 to obtain the correct value. Using the table of operator values produces:

```
        > Y = 3 + 5 * 9 <
VOP     0   1   2   3   0
```

After this, if the operations are performed in order of highest to lowest VOP value, the correct order is guaranteed.

A more complex example reinforces the point:

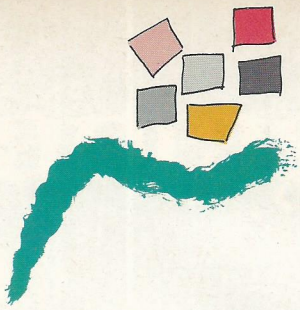
```
>Y=3*(4+20*6)*EXP(0.5)+9^2<
     +10     −10 +10 −10
0   1  3  12  13 3 100    2   5 0
```

The order of execution of the operators would again be from highest to lowest. However, to achieve this, multiple passes of the VOP values would have to be performed in a search for the next highest VOP value on each pass. (A more efficient, one-pass procedure will be discussed when the analysis table is presented.)

An important aspect of this VOP approach is the ease of checking for unbalanced parentheses. In the equation:

```
>Z = 14 * (3.14/2.0 + 4 <
        +10
0   1   3     13   12   10
```

there is no ending parenthesis and no subtraction of 10 from the next VOP values. Thus the ending operator has a value other than 0. If the VOP value for < is negative, then there are too many right-hand parentheses.



```
Program Parse;
var Found,KKQ,KL,KLL,JK,I,K,IncPar,L:Integer;
    Linel:String[8Ø];
    VOP:array[1..1ØØ] of Integer;
    VarList,VarEQ:array[1..1ØØ] of string[15];
    STl:String[15];
    Code,NumVar:Integer;
    TEO:array[1..1ØØ] of char;
    VarValue,RealVarEQ:array[1..1ØØ] of real;
Procedure CheckVar;
Begin
Found:=Ø;
For JK:=1 to NumVar do
    Begin
    If VarEQ[K]=VarList[JK] then
        Begin
{       This allows the left-hand variable to be used on the right side
        of the equation as well
}
        Found:=1;
        If K<>1 then RealVarEQ[K]:=VarValue[JK];
        end;
    end;
If (K=1) and (Found=Ø) then
    Begin
{   New variable found
    Initially set equal to zero
}
    Found:=1;
    NumVar:=NumVAr+1;
    VarList[NumVar]:=VarEq[K];
    VarValue[NumVar]:=Ø.Ø;
    end;
If Found=Ø then
    Begin
    Writeln('Error -- Variable ',VarEQ[K],' not found');
    Halt;
    end;
End;
Procedure Assignment;
Var Stl:String[15];
Begin
For JK:=1 to NumVar do
    Begin
    If VarEQ[1]=VarList[JK] then VarValue[JK]:=RealVarEQ[1];
    end;
end;
Procedure EvaluateTable;
Label Again,Again2;
Begin
While (TEO[1]<>'<') and (VOP[L]<=VOP[K]) do
Begin
{   Perform last operation    }

   Case TEO[K] of
       '+':RealVarEQ[K-1]:=RealVarEQ[K-1]+RealVarEQ[K];
       '-':RealVarEQ[K-1]:=RealVarEQ[K-1]-RealVarEQ[K];
       '*':RealVarEQ[K-1]:=RealVarEQ[K-1]*RealVarEQ[K];
       '/':Begin
           If RealVarEq[K]=Ø then
               Begin
               Writeln('Division by zero error ');
               Halt;
               end;
           RealVarEQ[K-1]:=RealVarEQ[K-1]/RealVarEQ[K];
           end;
```

Listing 1. *(Continued on following page)*

Now the VOP values are placed in a table and analyzed in such a fashion that only one pass of the equation is required. To see this, consider the following equation:

Z1 = 40 + (9 * 6) − 30

The analysis begins by finding the name of the variable to the left of the equal sign and the name of the equal sign itself:

L	Variable/ Value	Operator	VOP
1	Z1	>	0
		=	1

The analysis then continues because *VOP(L+1)* is greater than *VOP(L)*. This analysis produces:

1	Z1	>	0
2	40	=	1
		+	2

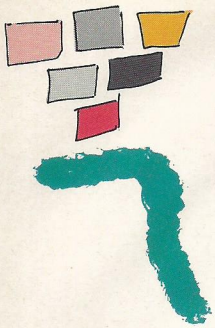
Since *VOP(3)* is greater than *VOP(2)*, the analysis continues:

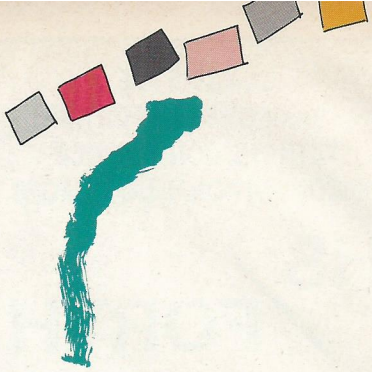
1	Z1	>	0
2	40	=	1
3	9	+	2
4	6	*	13
		−	2

At this point, *VOP(5)* is less than or equal to *VOP(4)*. This means that the previous operation can now be performed without upsetting the order of operators. Values needed or the operation to be performed are easily found in the table. *Value(L−1)* and *Value(L)* are the values used in the operation given in *Operator(L)*. This produces the new table:

1	Z1	>	0
2	40	=	1
3	54	+	2
		−	2

VOP(4) must now be compared to *VOP(3)* before the rest of the equation is processed. The addition operation can be then be performed, producing:





1	Z1	>	0
2	94	=	1
		–	2

Now the next term in the equation is added to the table since no other operation currently in the table can be executed. The table now looks like this:

1	Z1	>	0
2	94	=	1
3	30	–	2
		<	0

The last term, <, has a VOP of 0, which causes the minus operation to be performed:

1	Z1	>	0
2	64	=	1
		<	0

Now *VOP(3)* is less than *VOP(2)* and the assignment operation is performed. In other words, the value 64 is stored so that it can be referenced whenever the variable *Z1* is referenced. Another equation can now be processed.

Notice that only one pass of the equation is needed. The VOP values and analysis table are generated as the equation is scanned. Also, there is an increase in efficiency by performing a given operation as soon as possible (subject to the order of operators). The equation:

C=C+1

is easily handled if *C* has been previously defined.

Thus the parser includes the ability to use variables on the right side of the equal sign. The parser seeks out these previous definitions and substitutes the values in the table as these variables are found in the equations.

For a special function such as *EXP*, a one-letter symbol must be used to put this operator in the table. Otherwise, the analysis of an equation containing this operator is performed as previously explained.

```
          '^':RealVarEQ[K-1]:=Exp(RealVArEQ[K]*Ln(RealVareq[K-1]));
          '=':RealVarEq[K-1]:=RealVarEq[K];
          'E':RealVarEQ[K-1]:=exp(RealVArEQ[K]);
          '~':RealVarEq[K-1]:=-1*RealVarEq[K];
      End;
      VOP[K]:=VOP[K+1];
      TEO[k]:=TEO[K+1];
      K:=K-1;L:=K+1;
      If L<>1 then VarEQ[L]:='';
      If L<>1 then RealVarEQ[L]:=0.0;
end;
End;
Procedure Hier;
Begin
For K:=1 to 20 do
      Begin
      VarEq[K]:='';
      end;
For k:=1 to 100 do
      Begin
      VOP[K]:=0;
      end;
K:=1;
IncPar:=0;
For I:=1 to length(Linel) Do
      Begin
      L:=K+1;
      If Linel[I]='<' then Vop[L]:=IncPar;
      If (Linel[I]='+') or (Linel[i]='-') then Vop[L]:=2+IncPar;
      If Linel[I]='=' then VOP[L]:=1+IncPar;
      If Linel[I]='^' then VOP[L]:=5+IncPar;
      If Linel[I]='~' then VOP[L]:=200;
      If (Linel[I]='*') or (linel[I]='/') then VOP[L]:=3+IncPar;
      If Linel[I]='(' then
          Begin
          IncPar:=IncPar+10;
          Found:=0;
{  The following is set up in this fashion to allow for
   easy inclusion of other special functions   }
          If Vareq[K]='EXP' then Found:=1;
          If found=1 then
              Begin
              VOP[L]:=100+IncPar;
              TEO[L]:='E';
{             Symbol for EXP function                  }
              K:=K+1;
              L:=K+1;
              End;
      End;
      If Linel[I]=')' then IncPar:=IncPar-10;
      If Linel[I] in ['A'..'Z','a'..'z','0'..'9','.','~'] then
          Begin
          VarEq[K]:=VarEq[K]+Upcase(Linel[I]);
          end;
      If Linel[I] in ['+','-','=','*','/','^','<','~'] then
          Begin
          TEO[L]:=Linel[I];
        {  If VarEQ[K] contains a variable name, search variable list
           to find that variable.  If not found, error message.
           }
{
          All values are read in as strings, even numbers.
          These must be converted to real values by the VAL procedure.
```

Listing 1. *(Continued on following page)*

COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES

HS/FORTH

Now you can compile even large programs in the blink of an eye. If you don't need to compile the huge fast programs we handle so well, use the metacompiler to spin off compact ones — as small as a few hundred bytes for simple threaded utilities — as small as 2 kbytes for a full Forth execution core. HS/FORTH is the best base from which to spin off either direct or indirect threaded systems, small or large, or anything else you might invent. This is absolutely the most flexible Forth system available, at any price, with no expensive extras you'll need to buy later. Distribute metacompiled tools, or turnkeyed applications royalty free.

Although HS/FORTH is unmatched for language experimentation and development, remember that we wrote it to be a top notch application development system. Your application will have all of DOS at its disposal, commands, other programs, functions, everything! I/O is easier than in Pascal or Basic, but much more powerful — whether you need parsing, formatting, or random access. Send display output through DOS, BIOS, or direct to video memory. Windows organize both text and graphics display. Math facilities include both software and hardware floating point plus an 18 digit integer (finance) extension and fast arrays for all data types. The hardware floating point covers the full range of trig, hyper and transcendental math including complex. Forth gives you total control of your computer, but only HS/FORTH gives you implemented functionality so you aren't left hanging with "great possibilities" (and lots of work!)

We can't possibly cover everything in this ad, so please call or write and ask for our brochure. We'll also be happy to answer any questions.

HS/FORTH complete system:	$395.
Forth: Text & Reference (500pg) Kelly&Spies, Prentice Hall	$ 22.
HS/FORTH: Tutorial & Reference Kelly&Callahan (500pg)	$ 59.
HS/FORTH Glossary	$ 10.
DEMO DISK	$ 10.

VISA Visa Mastercard MasterCard

HARVARD
SOFTWORKS
PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

CIRCLE 25 ON READER SERVICE CARD

The program and extensions

The program presented in Listing 1 is a stand-alone version that can easily be included in an application program. In its present form, the program continues to request new equations until a variable *Q* is input.

As it stands now, this parser is useful for most applications, although more error checking could be incorporated. Also, other special operator functions could be easily added.

One other improvement would be to use the minus sign rather than the ~ for the unary minus operation. To do this, whenever a minus sign is found the previous character must be checked. If the previous character is an operator, then the minus sign represents the unary minus operation. Otherwise, the minus sign is just the subtraction operation.

References

1. Schildt, Herbert. *Advanced Turbo Pascal Programming and Techniques.* Berkeley, Calif.: Osborne McGraw-Hill, 1986.
2. Lee, John A.N. *The Anatomy of a Compiler,* 2nd ed. New York: D. Van Nostrand Co., 1974.

Scott Magruder is an assistant professor of economics at Austin Peay State University, Clarksville, Tenn. Currently he is directing a grant writing microcomputer tutorials for various business subjects.

Artwork: Mary Farr

```
}
        ST1:=VarEQ[K];
        If St1[1] in ['Ø'..'9'] then
            Begin
            If RealVarEQ[K]=Ø then Val(VarEQ[K],RealVarEQ[K],Code);
            If Code<>Ø then Writeln('Error in converting string to number');
            End;
        If St1[1] in ['A'..'Z','a'..'z'] then CheckVar;
        EvaluateTable;
        K:=K+1;
        End;
    If Linel[I]= '<' then
        Begin
        VOP[L]:=IncPar;
        If VOP[L]>Ø then Writeln('Error -- expected ) not found');
        If VOP[L]<Ø then Writeln('Error -- too many )s ');
        If VOP[L] <> Ø then Halt;
        end;
    end;
end;
Begin
NumVar:=Ø;
Repeat
For JK:=1 to 1ØØ do
    Begin
    VarEQ[JK]:='';
    RealVarEQ[JK]:=Ø.Ø;
    end;
    TEO[1]:='>';
    Write('give formula ');
    Linel:='';
    Readln(Linel);
    Linel:=Linel+'<';
    Hier;
    Assignment;


    For JK:=1 to NumVar do
        Begin
        Writeln('Var ',VarList[JK],'  Value  ',VaRValue[JK]);
        end;
Until UpCase(Linel[1])='Q';
end.
```

Listing 1. *(Continued from preceding page)*

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.
- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS®

SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA. 27511-8000. Telephone (919) 467-8000, x 7000

CL0687

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. Lattice is the registered trademark of Lattice, Inc. PLINK86 and PLIB86 are trademarks of Phoenix Software Associates Ltd. UNIX is the trademark of AT&T. Copyright © 1985 by SAS Institute Inc. Printed in the USA.

Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The MS/PC-DOS and XENIX software source that caters to your programming needs.

Discover the Many Advantages of Paradise...

- Lowest price guaranteed
- Latest versions
- Huge inventory, immediate shipment
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee*

NEW CLUES THIS MONTH!

Discover Paradise Contest— Search for the Hidden Clues to our Island's Identity and Win!

	LIST	OURS
C++		
ADVANTAGE C++	$ 495	479
PFORCE++	395	215
C COMPILERS		
AZTEC C-86 COMMERCIAL	499	CALL
AZTEC C-86 DEVELOPER	299	CALL
C86 PLUS	497	379
DATALIGHT C DEVELOPER'S KIT	99	79
DATALIGHT OPTIMUM C	139	109
LATTICE C 3.2	500	265
W/SOURCE	900	545
LET'S C	75	59
W/CSD DEBUGGER	150	99
MARK WILLIAMS C	495	289
MICROSOFT C 4.0	450	269
MICROTEK 68000/08/10/20 C CROSS	CALL	CALL
TURBO C	100	65
UNIWARE 68020 C CROSS	595	CALL
WIZARD C	450	309
C INTERPRETERS		
C-TERP	300	229
INSTANT C	500	379
INTERACTIVE C	249	219
INTRODUCING C	125	105
Q C	60	55
RUN/C	120	85
RUN/C PROFESSIONAL 1.1	250	155
ASSEMBLERS, LINKERS		
ADVANTAGE DISASSEMBLER	295	265
ADVANTAGE LINK	395	359
ASMLIB	149	129
DIS 'N DATA	165	CALL
MACRO ASSEMBLER	150	93
PASM 86	195	115
PLINK 86 PLUS	495	279
QUELO 68000 X-ASM	595	509

FREE SCIENTIFIC & ENGINEERING SOLUTIONS CATALOG

Describes 100 of the best software packages for solving complex equations, number crunching, analyzing data, 3-D CAD/CAM design, technical word processing, circuit design, sophisticated graphics and more. Call for your FREE catalog today!

	LIST	OURS
GRAPHICS		
ESSENTIAL GRAPHICS	250	195
GRAPHIC	350	289
GSS GRAPHICS DEVELOPMENT TOOLKIT	495	375
GSS KERNEL SYSTEM	495	375
GSS METAFILE INTERPRETER	395	235
GSS PLOTTING SYSTEM	495	375
HALO — ONE LANGUAGE	300	205
W/TEN FONT PACK	425	299
HALO — FIVE MICROSOFT LANGUAGES	595	389
METAWINDOW	185	115
METAWINDOW PLUS	235	185
METAFONTS	80	59
METAFONTS PLUS	235	185
C UTILITY LIBRARIES		
ASYNCH MANAGER	175	119
BASIC_C	175	129
BLACKSTAR FUNCTIONS	99	89
C ESSENTIALS	100	85
C-DISPLAY LIBRARIAN	99	79
C FOOD SMORGASBORD	150	95
W/SOURCE	300	179
C TOOLS PLUS	175	119
DISPLAY EXPRESS	99	79
ESSENTIAL C UTILITY	185	119
ESSENTIAL COMMUNICATIONS	185	125
ESSENTIAL COMMUNICATIONS PLUS	250	189
GREENLEAF COMM	185	125
GREENLEAF FUNCTIONS	185	125
INTERWORK	129	109
LINK & LOCATE	350	315
MULTI-C	149	135
MULTI-COMM	149	135
NUMBER OF ISLANDS	322	CALL
PFORCE	295	209
SYNERGY DEVELOPMENT TOOLKIT	395	249
TIMESLICER	295	265
W/SOURCE	1000	CALL
THE HAMMER	195	139
TOPVIEW TOOLBASKET	250	179
SCREEN DISPLAY, WINDOWS FOR C		
C WORTHY	295	269
CURSES	125	89
W/SOURCE	250	175
FLASH-UP WINDOWS	90	79
GREENLEAF DATA WINDOWS	225	155
W/SOURCE	395	289
MICROSOFT WINDOWS DEVELOPMENT KIT	99	65
ON-LINE HELP	500	309
PANEL	149	109
PANEL PLUS	295	215
POLYDESK III	495	395
QUICKSCREEN	99	79
SCREENPLAY (LATTICE)	295	265
SOFTSCREEN HELP	150	135
VIEW MANAGER	195	175
VITAMIN C 3.0	275	199
VC SCREEN	225	199
WINDOWS FOR C	99	85
WINDOWS FOR DATA	295	239
Z VIEW	395	315
	245	175
FILE MANAGEMENT		
ASMTREE	395	339
BTRIEVE	245	185
XTRIEVE	245	185
REPORT OPTION	145	99
BTRIEVE/N	595	455
XTRIEVE/N	595	455
REPORT OPTION/N	345	269
C-ISAM	225	195
C-TREE	395	315
R-TREE	295	239
C-TREE/R-TREE BUNDLE	650	519
CQL	395	329
DBC III/II	250	169
W/SOURCE	500	359
DBC III PLUS	750	599
W/SOURCE	1500	1195
DB_VISTA	195	145
W/SOURCE	495	399
DB_QUERY	195	155
W/SOURCE	495	399
FABS	150	129
FABS PLUS	195	169
INFORMIX	795	639
INFORMIX 4GL	995	799
INFORMIX SQL	795	639
LATITUDE	165	CALL
PHACT	295	265
DEBUGGERS		
ADVANCED TRACE 86	175	125
BREAKOUT	125	99
C-SPRITE	175	129
CODESMITH 86	145	105
CSD DEBUGGER	75	59
PERISCOPE I 3.0	345	295
PERISCOPE II 3.0	175	145
PERISCOPE II-X 3.0	145	109
PERISCOPE III	995	825
PFIX 86 PLUS	395	229
XVIEW 86	60	49
EDITORS		
BRIEF	195	CALL
DBRIEF	95	CALL
BRIEF/DBRIEF BUNDLE	275	CALL
C SCREEN	75	CALL
CVUE	75	59
W/SOURCE	250	195
EDIX	195	155
EMACS	295	265
W/SOURCE	995	895
EPSILON	195	155
FIRSTIME (C)	295	229
INHABITED ISLANDS	100	CALL
KEDIT	125	99
LSE	125	99
PC/VI	149	109
PMATE	195	115
SPF/PC	195	145
VEDIT	150	99
VEDIT PLUS	185	129
XTC	99	79
SORT UTILITIES		
AUTOSORT	150	129
MICROSOFT SORT	195	129
OPT-TECH SORT	149	115
PROFILERS		
CODE SIFTER	119	95
PFINISH	395	229
THE PROFILER	125	89
LINT		
PC LINT	139	99
PRE-C	295	155
MAKE UTILITIES		
ADVANTAGE MAKE	125	99
LMK	195	139
PMAKER	125	79
POLYMAKE V 2.0	149	119
SAPIENS MAKE	79	75

80386 DEVELOPMENT TOOLS

	LIST	OURS
386 ASM/LINK	495	399
HIGH C	895	759
PROFESSIONAL PASCAL	895	759
TRANSLATORS/BRIDGES		
BASTOC	495	399
BAS_C	85	79
BAS_PAS	85	79
DB2C	299	269
F2C	795	CALL
MB 86	895	CALL
PTRAL	179	CALL
R BRIDGE	395	339
TP2C	249	CALL
ARTIFICIAL INTELLIGENCE		
ARITY COMBO PACK	1095	979
ARITY PROLOG INTERPRETER	295	229
ARITY STANDARD PROLOG	95	79
GOLDEN COMMON LISP	495	CALL
GCL 286 DEVELOPER	1190	CALL
INSIGHT 2+	485	379
IQCLISP	300	CALL
IQLISP	270	CALL
MICROSOFT LISP	250	159
PROLOG 86 PLUS	250	CALL
TRANS LISP PLUS	195	CALL
TURBO PROLOG	100	65
TURBO PROLOG TOOLBOX	100	65
SMALLTALK V	99	85
BASIC		
BETTERBASIC	199	129
BETTERBASIC ADD ONS	CALL	CALL
BETTERTOOLS	95	89
87 QB PAK	69	59
87 SOFTWARE PAK	180	149
EXIM TOOL KIT	50	49
FINALLY	99	89
FINALLY MODULES	99	89
GDPEDIT	99	89
INSIDE TRACK	65	55
MACH 2	75	59
MICROSOFT QUICKBASIC	99	65
PEEKS 'N POKES	45	39
PROFESSIONAL BASIC	99	75
8087 MATH SUPPORT	50	45
PANEL-BASIC	145	115
QUICK PAK	69	59
STAY RES	95	75
TRUE BASIC	150	105
ADD ONS	CALL	CALL
TURBO BASIC	100	65
COBOL COMPILERS/UTILITIES		
FPLIB	149	129
MICROSOFT COBOL	700	439
MICROSOFT SORT	195	129
MICRO/SPF	175	155
OPT-TECH SORT	149	115
REALIA CICS	995	785
REALIA COBOL	995	785
RE`LIA MENU	150	119

We'll Match Any Nationally Advertised Price.

	LIST	OURS
RM/COBOL	950	649
RM/COBOL 8X	1250	909
SCREENIO	400	CALL
SCREENPLAY	175	129
SPII	345	279
VISUAL COBOL (MBP)	795	695
FORTRAN COMPILERS/UTILITIES		
ACS TIMES SERIES	495	389
DIAGRAM'ER	129	115
DOCUMENT'ER	129	115
87 SFL	250	225
FORLIB 2.0	95	85
FORLIB-PLUS	70	45
FORTRAN SCIENTIFIC SUBROUTINES	295	249
FOR-WINDS	90	69
GRAFLIB 2.0	175	159
GRAFMATICS OR PLOTMATICS	135	119
GRAFMATICS AND PLOTMATICS	240	219
LAHEY FORTRAN	477	CALL
MICROSOFT FORTRAN 4.0	450	269
NO LIMIT	129	115
PLOTHI 2.0	175	159
PLOTHP 2.0	175	159
RM/FORTRAN	595	389
STRINGS AND THINGS	70	55
MODULA-2		
FARBWARE MODULA-2	90	79
LOGITECH MODULA-2		
APPRENTICE PACKAGE	99	79
WIZARD'S PACKAGE	199	159
MAGIC TOOLKIT	99	79
ROM PACKAGE & RT DEBUGGER	299	239
WINDOW PACKAGE	49	39
REPERTOIRE	89	75
PASCAL COMPILERS		
MARSHALL PASCAL	189	169
MICROSOFT PASCAL	300	185
PASCAL-2	350	329
TURBO PASCAL	100	65
NUMERICAL METHODS TOOL BOX	100	65
OTHER BORLAND	CALL	CALL
TOOLS FOR TURBO PASCAL		
ALICE	95	69
FIRSTIME	75	59
FLASH-UP WINDOWS	90	79
SCREENPLAY	100	89
SCREEN SCULPTOR	125	95
T-DEBUG PLUS	60	50
TURBO•ASM	100	85
TURBO DATABASE TOOLBOX	70	45
TURBO EDITOR TOOLBOX	70	45
TURBO EXTENDER	85	65
TURBO GRAPHIX TOOLBOX	70	45
TURBO HALO	129	99
TURBO PASCAL ASYNCH MGR	100	79
TURBO PROFESSIONAL	70	49
TURBO POWER TOOLS PLUS	100	79
TURBO REF	50	45
TURBO SMITH	69	45
TURBO WINDOWS	80	65
OTHER TURBO TOOLS	CALL	CALL
OTHER LANGUAGES		
IF YOU DON'T SEE IT—CALL!!!!!		
APL*PLUS	595	429
JANUS/ADA C PACK	95	89
JANUS/ADA D PACK	900	769
JANUS/ADA ED PACK	395	CALL
LATTICE RPG II COMPILER	750	629
SCREEN DESIGN AID	350	309
SOURCE ENTRY UTILITY	250	199
SORT/MERGE UTILITY	250	199
RPG COMBO	1100	879
PC FORTH	150	109
PERSONAL REXX	125	99
POCKET APL	95	75

	LIST	OURS
DBASE COMPILERS, RELATED PRODUCTS		
CLIPPER	695	459
DBC III PLUS	750	599
DBRIEF	95	CALL
FOXBASE	395	CALL
FOXBASE PLUS	395	CALL
GENIFER	395	309
QUICK SILVER	599	499
TOM RETTIG'S LIBRARY	100	79
DIGITAL RESEARCH		
CB 86	600	569
PASCAL MT+	400	379
PL1—86	750	709
DR ASSEMBLER + TOOLS	200	189
ACCESS MANAGER	400	379
DISPLAY MANAGER	500	475
FORTRAN 77	350	329
BLAISE		
ASYNCH MANAGER	175	119
C TOOLS PLUS	175	119
EXEC	95	69
PASCAL TOOLS	125	99
PASCAL TOOLS 2	100	85
PASCAL TOOLS & PASCAL TOOLS 2	175	119
RUNOFF	50	45
TURBO ASYNCH MANAGER	100	79
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER	275	179

NEW PRODUCTS

Microport System V/AT—New release 2.2 of complete system has numerous enhancements, faster speed, improved documentation. Includes free Nutshell guide to UNIX and Sysvision, a user-friendly system administrator front end to UNIX.
List $549 **Ours $465**

Microsoft COBOL v. 2.2—Latest release of GSA-certified COBOL compiler. Now includes Microsoft COBOL TOOLS with ViewCob, interactive symbolic debugger and COBOL cross-reference generator.
List $700 **Ours $439**

Periscope III—Real-time hardware breakpoint debugger. Find errors in real-time systems, stop intermittent failures, interface with undocumented systems and eliminate bottlenecks in your code. Includes board, break-out switch, software and manual.
List $995 **Ours $825**

Turbo C—Long-awaited C compiler from Borland. Complete interactive development environment. ANSI C compatible; includes editor, MAKE, pull-down menus and windows.
List $100 **Ours $65**

	LIST	OURS
LATTICE		
LATTICE C COMPILER	500	265
W/SOURCE	900	545
C CROSS REFERENCE GENERATOR	50	39
C FOOD SMORGASBORD	150	95
W/SOURCE	300	179
C-SPRITE	175	129
CURSES	125	89
W/SOURCE	250	175
DBC III/II	250	169
W/SOURCE	500	359
DBC III PLUS	750	599
W/SOURCE	1500	1195
LMK	195	139
SECRETDISK	120	89
SIDETALK	120	89
SSP/PC	350	269
TEXT MANAGEMENT UTILITIES	120	89
TOPVIEW TOOLBASKET	250	179

	LIST	OURS
LIFEBOAT		
ADVANTAGE C++	495	479
ADVANTAGE DISASSEMBLER	295	265
ADVANTAGE LINK	395	359
ADVANTAGE MAKE	125	99
PANEL PLUS	495	395
QUICKSCREEN	295	265
RUN/C	120	85
RUN/C PROFESSIONAL 1.1	250	155
TIMESLICER	295	265
W/SOURCE	1000	CALL
MICROSOFT		
C COMPILER	450	269
COBOL	700	429
FORTRAN	450	269
LISP	250	159
MACRO ASSEMBLER	150	93
MUMATH	300	185
PASCAL	300	185
QUICKBASIC	99	65
SORT	195	125
WINDOWS	99	65
WINDOWS DEVELOPMENT KIT	500	309
PHOENIX		
PASM 86	195	115
PDISK	145	89
PFANTASY PAC	995	599
PFINISH	395	229
PFIX 86 PLUS	395	229
PFORCE	395	209
PFORCE++	395	215
PLINK 86 PLUS	495	279
PMAKER	125	79
PMATE	195	115
PRE-C	295	155
PTEL	195	115
POLYTRON		
C BEAUTIFIER	49	45
C LIBRARY I	99	79
POLYBOOST	80	69
POLYDESK III	99	79
POLYDESK III ADD ONS	CALL	CALL
POLYLIBRARIAN	99	79
POLYLIBRARIAN II	149	115
POLYMAKE 2.0	149	119
POLYSHELL	149	119
POLYXREF	219	169
POLYXREF—ONE LANGUAGE	129	99
POWERCOM	179	139
PVCS PERSONAL	149	119
PVCS CORPORATE	395	309
PVCS NETWORK	1000	CALL
OPERATING SYSTEMS		
OPERATING SYSTEM		
TOOLBOX (WENDIN)	99	79
PCNX	99	79
PCVMS	99	79
SYSTEM V/AT (MICROPORT)	549	465
DOS MERGE 286	149	125
RUNTIME SYSTEM	199	169
SOFTWARE DEV. SYSTEM	249	209
TEXT PREPARATION SYSTEM	199	169
UNLIMITED USER UPGRADE	249	209
XENIX SYSTEM V (SCO)	1295	999
XENIX DEV. SYSTEM	595	499
XENIX OPERATING SYSTEM	595	499
XENIX TEXT PROC. PKG.	195	145
DESKTOP PUBLISHING		
HALO DPE	195	155
XEROX VENTURA PUBLISHER	895	695
TURBO VENTURA PUBLISHER	1595	1249

XENIX/UNIX PRODUCTS

	LIST	OURS
ADVANTAGE C++	695	CALL
BTRIEVE	595	465
C-ISAM	319	285
C-TERP	498	CALL
C-TREE	395	329
R-TREE	395	329
DB_VISTA	CALL	CALL
FOXBASE	795	CALL
INFORMIX	995	795
INFORMIX 4GL	1500	1235
INFORMIX SQL	995	795
LYRIX	595	449
MICROSOFT BASIC	350	239
MICROSOFT COBOL	995	629
MICROSOFT FORTRAN	695	439
MICROSOFT PASCAL	695	439
NETWORKS FOR XENIX	595	495
PANEL	625	535
RM/COBOL	1250	949
RM/FORTRAN	750	549
SCO PROFESSIONAL	795	595
UNIWARE 68020 C CROSS	1390	CALL

	LIST	OURS
TELECOMMUNICATIONS		
ANNUAL RAINFALL (MM)	3000	CALL
ASCOM IV	195	175
PTEL	195	115
SIDETALK	120	89
ADDITIONAL PRODUCTS		
C TOOL SET	95	CALL
COMMAND PLUS	80	69
DAN BRICKLIN'S DEMO PROGRAM	75	59
FASTBACK	175	135
HELP/CONTROL	125	99
INTERACTIVE EASYFLOW	150	129
LATTICE TEXT MGMT. UTIL.	120	89
MICROSOFT MACH 10 W/MOUSE & WINDOWS	549	379
MKS TOOLKIT	139	CALL
NORTON COMMANDER	75	55
NORTON UTILITIES	100	59
PC/EDT	250	CALL
PDISK	145	89
SAPIENS V8	300	269
SCREEN MACHINE	79	59
SCROLL & RECALL	69	59
SECURITY LIBRARY	125	CALL
SOURCE PRINT	75	69
TASKVIEW	80	59
TLIB	100	89
TREE DIAGRAMMER	55	50
VTEK	150	129
ZAP	95	CALL

SOFTCRAFT SPECIALS

	LIST	OURS
BTRIEVE	245	185
XTRIEVE	245	185
REPORT OPTION	145	99
BTRIEVE/N	595	455
XTRIEVE/N	595	455
REPORT OPTION/N	345	269

CALL TOLL FREE: **1-800-445-7899**

In NY: 914-332-4548

Terms and Policies

- **We honor MC, VISA, AMERICAN EXPRESS**

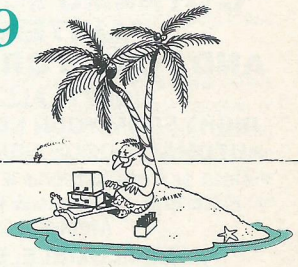
No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling $3.00 per item, sent UPS ground. Rush service available, prevailing rates.

- Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
- **Mention this ad when ordering**—some items are specially priced.
- Prices and Policies subject to change without notice.
- Corporate and Dealer inquiries welcome.

*Ask for details. Some manufacturers will not allow returns once disk seals are broken.

Programmer's Paradise

42 River Street, Tarrytown, NY 10591



DISCOVER PARADISE CONTEST

Win The High-Performance Software Development Package, Valued At Over $2,000!

Now that you've seen what Programmer's Paradise looks like, we want you to guess where it is! Just search this ad for clues to the identity of our tropical island paradise (Hint: They're mixed in with our product listings). Then send in your guess on this coupon. You could be the lucky winner of the high-performance software development package**, featuring: • **Univation's Dream Board**—Multifunction accelerator board with EMS. Designed to fit into a single expansion slot. The Dream Board combines the functions of three of the most popular enhancement boards—multifunction, EMS, accelerator—into one power-packed package • **ADVANTAGE Link** overlay linker • **Microsoft** compiler of your choice (MS/PC-DOS) • **Editor** of your choice (binary). **... VALUE: OVER $2,000**

UNIVATION System Solutions For Business

LIFEBOAT Microsoft®

Complete this form and mail to: DISCOVER PARADISE CONTEST, c/o Programmer's Paradise, 42 River Street, Tarrytown, NY 10591. Entries must be received by August 1, 1987.

OFFICIAL RULES: 1. Limit one name per entry, no purchase necessary. Enter as often as you wish, mailing each entry on a separate postcard. All entries must be received by August 1, 1987. 2. Participants must be 18 years or older and a resident of the United States. Void where prohibited by law. 3. Winner will be responsible for all taxes. 4. Odds of winning will be determined by number of entries received. Winner will be selected out of all correct entries received. 5. Prize will be awarded within 30 days of all entries received. **No substitutions allowed.

I think the name of the island in your ads is: _____

Name _____ **Title** _____

Company name _____

Company address _____

City _____ **State** _____ **Zip** _____

Phone # _____

CIRCLE 26 ON READER SERVICE CARD

C POWER

FREE! Turbo C® (Borland)

(with C Business Library or C Starter Package until 6/30/87.)

POWER WINDOWS
FUNCTION LIBRARIES
SUPERFONTS FOR C
B-TREE & ISAM
C-TERP
SCREEN PAINTER
& FORMS CREATOR

POWER WINDOWS

MOST POWERFUL YET
POP-UP WINDOWS FOR
Menus/Overlays
Help Screens
Messages/Alarms
ZAP ON/OFF SCREEN
FILE-WINDOW MANAGEMENT
AUTOMATIC
Horizontal & Vertical Scrolling
Word Wrap & Line Insertion
Cursor/Attributes/Borders

The most powerful, flexible and easy to use windowing package available! Many types of menus. Highlighting. Move data between files, keyboard, program and windows. Status lines. Change size/location/overlapping. Move/add/delete/cascade windows. Full source code. No royalties.
3 disks $129.95

C FUNCTION LIBRARY

BEST YOU CAN GET
400 FUNCTIONS
FULLY TESTED
SUPERB DOCUMENTATION
BETTER FUNCTIONS

Most complete screen handling plus graphics; cursor/keyboard/data entry, 72 string functions with word wrap; status and control; utility/DOS BIOS/time/date functions; printer control & more. Special functions. Functions you NEED but don't have! Full source code. No royalties.
4 disks $129.95

C BUSINESS LIBRARY

INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C, B-TREE LIBRARY, ISAM
ALL for $299.95
(A $440 VALUE)

SUPERFONTS FOR C

SUPER SIZE CHARACTERS
Monochrome adapter
Color/graphics adapter
8 FONT LIBRARIES

Dramatic high impact screens/titles/messages. Use with or without windows. Great for projection TV. Create own font & image libraries or use ours. Use with windows for special effects or animation. Full source code. No royalties.
Font and Function Library . . .$49.95

B-TREE LIBRARY & ISAM DRIVER

POWERFUL DATA MANAGER
Fixed/Variable length records
FAST! EASY TO USE!
16.7 MILLION RECORDS/FILE
16.7 MILLION KEYS/FILE

Fast B-tree indices. Add/remove keys Find first/last/next/any key. Find keys by Boolean selection. Read/write/delete or add records to file. Full source. No royalties $129.95

C STARTER PACKAGE

INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C
ALL for $199.95
(A $310 VALUE)

COMPATIBILITY: *IBM PC/XT/AT & compatibles. Microsoft C 4.0, Lattice, Mark Williams, Aztec, CI 86, Datalight, Ecosoft, any ANSI or full K&R compiler.*

DEPTH: *Screen Painter/Data Entry/Power Windows (the most powerful professional windowing package) and C Function Library (the most extensive and useful library available) plus the speed, power and flexibility of the B-TREE and ISAM library. All work together to make your project go FAST!*

NO MATTER WHAT ELSE YOU HAVE, GET THESE...
DON'T PAY MORE AND GET LESS!

C DYNAMO SCREEN PAINTER AND FORMS CREATOR

DOES IT ALL
RIGHT FROM YOUR KEYBOARD
AUTOMATIC CODE GENERATION
Data or Help Screens & Windows
Data Entry Screens & Windows
Menus
FAST, FLEXIBLE, EASY
Save Man-Months of Programming
Full Control of Screen Attributes
Monochrome or Color
by Form, Screen, Window or Item
FAST EASY MENU GENERATION
1-2-3 Like, Many Others

Requires Dynamo Data Entry
Screen painter & manual $129.95

DYNAMO DATA ENTRY

UNIQUE, POWERFUL, NECESSARY
Full Validation of Each Keystroke
Range Checking
Data Security to Item Level
Scrollable Data Entry Forms with Fixed & Scrollable Parts
Allows Forms Larger Than Screen
Over 40 Item Types
Powerful "Picture" Capability
Unique: Mix Text, Data Entry Fields With Menu Items

Full source code. No Royalties
Code plus manual $129.95

C-TERP

INTERPRETER FOR C
NO COMPROMISE, FULL K&R
BUILT IN SCREEN EDITOR
FAST, FAST COMPILE/LINK
USE EXTERNAL LIBRARIES!
SYMBOLIC DEBUGGING
SINGLE STEPPING
RAVE REVIEWS!

2 disks and manual. $299.95
With and library $199.95

C TOTAL LIBRARY

INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C, B-TREE LIBRARY, ISAM AND C-TERP C interpreter
ALL for $499.95
(A $740 VALUE)

	LIST
PCNX™ UNIX WORKALIKE	$99
PCVMS™	99
O/S TOOL BOX™ (Build your own operating system)	99

™WENDIN SOFTWARE

THE C POWER FAMILY ... YOUR COMPLETE SOURCE FOR C

Entelekon™

SINCE 1982

12118 Kimberley, Houston, TX 77024 713-468-4412 VISA-MASTERCARD-CHECK-COD

CIRCLE 51 ON READER SERVICE CARD

A Source-Level Symbolic Debugger for UNIX

By David Livshin

Traditional software product development involves repeated execution of the edit/compile/link/debug chain. In such an environment, changing a program may be time- and resource-consuming.

One approach that has become popular recently uses interpreters for software development.[1] This solution has the virtue of being interactive, which speeds development, but you end up paying enormous time penalties when using interpreted code.

In addition to its standard debugger features (breakpoints, memory view/modification, etc.), ddb, an interactive source-level symbolic debugger, possesses the ability to execute C code on line. It debugs the compiled object code and, in cases where the source should be changed, extracts the changed code from the object file and interprets the new source instead. This process allows high-speed object file execution as well as the ability to change a program to a low-penalty operation requiring no recompilation or relinking. In addition, C constructs (expressions, flow control, etc.) are incorporated into the debugger commands, allowing more powerful flow control.

ddb description

ddb is implemented for UNIX systems.[2] It can be used to examine object and core files and provides a controlled environment for their execution. The object file is in Common Object File Format (COFF) and is normally created by a C compiler with the $-g$ option turned. However, ddb may be used with object files of programs written in any language, with or without debugger information present.

ddb has two working modes: sdb and C. In sdb mode, ddb behaves as a slightly extended sdb symbolic debugger and has all sdb features.

In C mode, the ddb command is a sequence of C language statements. These statements may refer to symbols of the debugged object file and may introduce new symbols. Statements in C mode may use sdb commands as escape sequences.

sdb mode

sdb is a general-purpose debugger that allows interaction with the debugged program on source language and machine levels. It provides a rich variety of commands, such as modifying and viewing memory, breakpoints, and on-line procedure execution. Since the sdb debugger is well documented,[2] I will give only a brief explanation of its additional commands. Table 1 gives a more detailed explanation of the commands used in this article and in the program examples.

Every new command has a two-character mnemonic that distinguishes it from sdb commands and makes its name more meaningful.

History. ddb maintains a history list into which it places every entered command. The number of commands available from this list is controlled by the built-in variable *_HIST_SZ_*, which may be changed in C mode. The commands that access and manipulate the history list allow you to print the existing

ddb commands

addr (b) [commands]	Set breakpoint at the specified address. commands (if specified) are executed when breakpoint is encountered and execution continues. If *k* is used as a command to execute at breakpoint, control returns to ddb instead of continuing execution.
c	Continue after a breakpoint or interrupt.
linenumber g	Continue after a breakpoint with execution resumed at the given line.
i	Single step by one machine instruction.
mc	Set ddb working mode to C mode.
m0	Set ddb working mode to sdb mode.

Table 1.

history list, reexecute the command from the history list, and edit the command from the history list. Editing is done using a line editor with a set of commands similar to the EMACS editing mode of the Korn shell.[3]

Macros. Macros are implemented by making a history event permanent (it would be kept in the history list even if more than *_HIST_SZ_* commands are entered after this event). The commands that support macros allow you to create and delete macros and print a list of active macros.

Memory check. This command accepts as an argument initial address and length of the memory region to be checked. It resumes execution in the single-instruction mode until at least one of the locations in the specified region is modified.

Switch modes. These commands enable establishment of the debugger working mode—either sdb or C.

C mode

The ddb command in this mode is a sequence of C language statements. It might be thought that these statements are the body of the block that is inserted in the program right after the position the program pointer points to. Thus, executing:

```
statement 1 ... statement n
```

is equivalent to executing the program:

```
   counter
{
statement 1 ... statement n
}
debugged program after program
   counter
```

starting with the entrance to the newly introduced block.

If a statement starts with $, ddb assumes that all the text up until the semicolon or the end of a line—whichever comes first—is an sdb command.

Being evaluated in the context of the stopped program, ddb commands treat symbols the same way C does. This means that if a symbol is acceptable from the inserted block, ddb treats it as a corresponding C symbol; otherwise, ddb creates a local symbol with the type *long*.

Program examples

When involved, ddb starts its work in the sdb mode, which means the user may debug a program exactly as he or she will do it with the sdb debugger. This allows the execution of debugger commands (setting and deleting breakpoints, running the program in different ways, displaying memory by different formats, examining the source, etc.). To transfer to C mode, *mc* should be executed.

Let's assume we want to debug the program shown in Listing 1. Note that in the debugging scripts shown in Listings 2, 3, and 4, all user input appears in bold and is highlighted, all terminal output appears in regular typeface, and comments are enclosed in semicolons. Note also that examples were computed on a 68000-based computer.

Code patching

Let's say we want to change the assignment:

```
x[i].x_m.y[j] = i*10000 + j;
```

on line 22 of Listing 1 to:

```
x[i].x_m.y[j] = i + j;
```

when *i* is 1 and *j* is 0 or when *i* is 5 and *j* is greater than 1 (Listing 3). This example demonstrates the fundamental difference between an interpreter and ddb.

```
 1.
 2.      #include <stdio.h>
 3.
 4.      main()
 5.       {
 6.        struct
 7.         {
 8.          long y_m;
 9.          struct
10.           {
11.            long y[6];
12.           } x_m;
13.         } x[6];
14.        long ar[6][6];
15.        long i, j;
16.
17.        for( i = 0; i < 6; i++ )
18.         {
19.          x[i].y_m = i*100000;
20.          for( j = 0; j < 6; j++ )
21.           {
22.            x[i].x_m.y[j] = i*10000 + j;
23.            ar[i][j] = i*1000 + j;
24.           }
25.         }
26.        for ( i = 0; i < 6; i++ )
27.         {
28.          for ( j = 0; j < 6; j++ )
29.           printf( "\t%d", x[i].x_m.y[j] );
30.          printf( "\n" );
31.         }
32.
33.       } /*   main  */
```

Listing 1.

```
Simple example

$ ddb tst          ; debugging program `tst' by ddb ;
1 ddb>main:b       ; setting breakpoint at `main' ;
                     132 -          main+12    [tst.c:17]
2 ddb>r            ; begin program execution ;
Debugging process 851.
bpt/trace at
132   |  main+12 [tst.c:17]   17:    for( i = 0; i < 6; i++ )
3 ddb>mc           ; entering C mode ;
    ; we are going to write a C program to execute 6 machine instructions ;
tmp = 0;           ; `tmp' is not acceptable from the `main'--assuming ;
                     ; it is not external--therefore it would be ;
                     ; created by ddb as a local symbol and will ;
                     ; disappear after exiting the created block ;
while( tmp < 6 )
 {
  $i               ; Note how sdb command is used in C mode ;
  tmp += 1;
 }
^D                 ; terminates input of the C mode command ;
   ; following is output of the i command ;
       [tst.c:19]
main+14:          0x86:          MOVE    i,D2
main+16:          0x88:          MULU    #$1c,D2
main+20:          0x8C:          LEA     (-168,FP),A2
main+24:          0x90:          MOVE.L  i,D1
main+26:          0x92:          MOVE.L  #0x186A0,D0
main+32:          0x98:          JSR     lmult
4 ddb>$mo          ; switch to sdb mode ;
5 ddb>c            ; continue execution ;
  ; output of the program that we debugged ;
        0        1        2        3        4        5
        10000    10001    10002    10003    10004    10005
        20000    20001    20002    20003    20004    20005
        30000    30001    30002    30003    30004    30005
        40000    40001    40002    40003    40004    40005
        50000    50001    50002    50003    50004    50005
Process 851 terminated.
6 ddb>q       ; exit ddb ;
```

Listing 2.

```
$ ddb tst
1 ddb>22:b mc       ; setting breakpoint at line 22 and specifying ;
                      ; command to be executed when breakpoint is reached ;
if ( ( ( i == 1 ) && ( j == 0 ) ) || ( ( i == 5 ) && ( j > 1 ) ) )
 {
  x[i].x_m.y[j] = i + j;    ; New code to be executed ;
  $23 g      ; Continue execution from line 23, thus extracting line 22 ;
             ; and replacing it by the new code ;
 }
^D
     164 -     main+44 [tst.c:22]
2 ddb>r            ; begin program execution ;
Debugging process 886.
        0        1        2        3        4        5
        1        10001    10002    10003    10004    10005
        20000    20001    20002    20003    20004    20005
        30000    30001    30002    30003    30004    30005
        40000    40001    40002    40003    40004    40005
        50000    50001    7        8        9        10
Process 886 terminated.
4 ddb>q
```

Listing 3.

Magus Inc. **Screen Designer**

Data&Windows

Data&Windows, the window-oriented data-entry system for Magus, Inc., now supports more of your favorite compilers! If you use Microsoft C, Pascal, or FORTRAN, Manx Aztec C86, Mark Williams C Programming System, Lattice C, IBM C, MetaWare High C, MetaWare Professional Pascal, or Datalight Optimum C, you can use Data&Windows to quickly and easily create almost any user interface.

With Data&Windows, you draw your text, fields, and colors on the video display. You see exactly what you will get while you create it. The screen designer gives you maximum functionality in an editor-like environment. Fields have options for specifying character and data type validations, user-definable validations, protected (display only) fields, auto-erasing fields, retain data fields, and more. You can check the look and operation of the screen with test mode.

Screens are saved in Microsoft object file format. You simply link the screen files with your application, so screen and field information is stored in your program (.EXE) file rather than in separate data files that must be present at runtime.

You get library routine support for fully overlapping and scrollable windows which use an approach called **static windowing**. This technique eliminates the need for replication of static data in dynamic memory. More than seventy library routines are available to manipulate your windows, control data entry and storage, create window-oriented menus, and more!

Get Started Quick! A thorough on-disk tutorial is provided so that you can begin creating useful screens the day you receive the product. Utility programs that document each screen and allow you to prototype (or simulate) your application complete the package.

Try It Out! Order your demo disk today. For $5.00 (refundable when you buy the product), you will receive a copy of the screen generator, the tutorial, and on-disk documentation on the utility programs and library routines.

Data&Windows is $345, or $695 with library source code, and certain discounts are available. Be sure to specify which compiler(s) you use when ordering.

Call (713) 665-4109 for more information. Major credit cards accepted.

REQUIREMENTS:
IBM PC/XT/AT/JR or true compatible, DOS 2.0 or later, at least 128K free RAM, and one of the compilers listed above. Xenix support will soon become available.

IBM, IBM PC, IBM XT, and IBM AT are trademarks of International Business Machines. Microsoft and XENIX are trademarks of Microsoft Corporation. Aztec is a trademark of Manx Software Systems. High C is a trademark of MetaWare.

Send to:
MAGUS, INC.
4545 Bissonet Suite #114
Bellaire, TX 77401

CIRCLE 28 ON READER SERVICE CARD

The program will be executed at the full speed of regular object code. Only when a specified breakpoint is reached (the breakpoint at line 22) will the code associated with this breakpoint be interpreted.

Conditional breakpoints. In many cases we wish to stop the program at a certain location only when some predicate is fulfilled. The usual sdb commands provide little help in solving such a problem (especially when the location is inside the large loop and/or the predicate is complex). The example in Listing 4 demonstrates how conditional breakpoints may be implemented in ddb. It also shows how C language constructs may be combined with the sdb commands.

Let's assume we want to stop at line 23 of our sample program:

```
1 ddb>23b mc
if ( ( j >= 5 ) && ( x[i].y_m >= 400001 ) )
 {
  $k  ; Stop execution and return control to ddb ;
 }
else
 {
  if ( ( i + j ) >= 8 )  ; check if we need to print values of ;
                         ; i, j, and x[i].y_m ;
  {
   printf( " i = %d  j = %d x[%d].y_m = %d\n", i, j, i, x[i].y_m );
  }
 }
^D

              202 -     main+82 [tst.c:23]
2 ddb>r
Debugging process 1002.
   ; output of the new code ;
 i = 3 j = 5 x[3].y_m = 300000
 i = 4 j = 4 x[4].y_m = 400000
 i = 4 j = 5 x[4].y_m = 400000
 i = 5 j = 3 x[5].y_m = 500000
 i = 5 j = 4 x[5].y_m = 500000
bpt/trace at
202  |  main+82 [tst.c:23]   23:       ar[i][j] = i*1000 + j;

3 ddb>c
 ; output of the program that we debugged ;
        0      1      2      3      4      5
       10000  10001  10002  10003  10004  10005
       20000  20001  20002  20003  20004  20005
       30000  30001  30002  30003  30004  30005
       40000  40001  40002  40003  40004  40005
       50000  50001  50002  50003  50004  50005
Process 1002 terminated.


4 ddb>q
```

Listing 4.

```
ar[i][j] = i*1000 + j;
```

but only when $j >= 5$ and $x[i].y\_m >= 400001$;. Otherwise, we want to see values of *i, j,* and $x[i].y\_m$ for *i* and *j* such that $i+j \geq 8$.

What makes ddb different

In this section we look at the features that distinguish ddb from other available debuggers.

Easy to master. The traditional UNIX system development environment for a C programmer involves numerous tools, each of which implies a different language (Shell, C, Make, etc.). This can lead to a large amount of time spent learning them. Consequently, some programmers will handle only part of the language and won't explore each tool to its fullest extent.

This problem does not exist in the ddb approach. ddb is designed to best suit the needs of the C programmer. Accepting C language statements as commands allows a C programmer familiar with sdb to master ddb fast and efficiently.

Powerful. ddb uses the C language for constructing debugger commands as well as patching the source being debugged; the patching actually becomes a debugger command, which leads to constructions of remarkable power and flexibility. The most outstanding features are complete flow control, on-line code execution, and code patching.

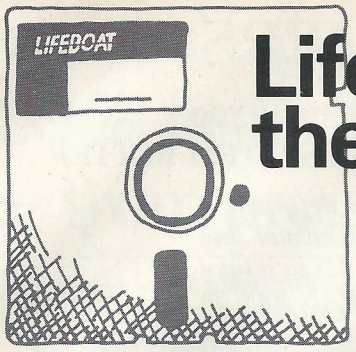
Fast. Because the program being debugged is executed in the object code, there is no slowdown in execution time. This allows the combination of the best of two worlds in one product: fast turnaround for modifications (which is offered by interpreters) and full execution speed (which interpreters lack).

References

1. Feuer, Alan R. "si—An Interpreter for the C Language," *USENIX Conference Proceedings,* pp. 47-55, June 1985.
2. AT&T. *UNIX System V Programming Guide.* April 1984.
3. Kochan, Stephen G., and Patrick H. Wood. *UNIX Shell Programming.* Hayden Book Co., 1985.

David Livshin is a software engineer with over 12 years' experience in software design and development.

Artwork: Marge Collins



Lifeboat. Your best source for the best names in software.

ADVANTAGE C++

This new object-oriented language lets you develop large and complex programs with greater resilience, fewer bugs. Write reliable, reusable code that is easier to understand and maintain. Fully compatible with existing C programs and tools. All the benefits of C without its limitations. Available for Lattice and Microsoft C compilers; MS-DOS and XENIX operating systems.

Latest From LIFEBOAT

Lattice C Compiler

The latest update of the ideal tool for developing high-performance MS-DOS applications in C. Full implementation of K&R C with UNIX and ANSI extensions. Offers the widest selection of C support tools. Now with Microsoft Windows support.

Lattice

Microsoft FORTRAN

New version produces smallest, fastest PC code. Uses the same optimizer and code generator from the Microsoft C compiler. Full ANSI FORTRAN 77 compatibility. GSA-certified to be error-free at the highest level. Includes advanced intrinsic math functions, Microsoft's CodeView debugger, a MAKE utility, linker, library manager and many other enhancements.

Microsoft

TimeSlicer

A library of C functions to create multitasking and real-time programs at the application level rather than interfacing with the operating system. Create, suspend or terminate tasks at run-time. Compatible with Lattice C, Microsoft C, ADVANTAGE C++ and object-oriented programming.

Latest From LIFEBOAT

ADVANTAGE Link

The fastest, most powerful PC-DOS linker available, and the first to take full advantage of EMS. Accepts Microsoft and Phoenix command files, and is compatible with Microsoft CodeView.

Brand New LIFEBOAT

VEDIT PLUS

For years, thousands of programmers have depended on VEDIT. If you take your editing seriously, take a good look at the all-new VEDIT PLUS.
Open windows to simultaneously edit several files. Access editing functions with pop-up menus. Use keystroke macros to speed editing. And run other programs within VEDIT PLUS—all with uncompromising speed, flexibility and power. VEDIT PLUS is completely customizable and available for MS/PC-DOS, CP/M 80/86. Ask for your FREE demo/tutorial disk.

CompuView

Greenleaf Data Windows

A new concept C library containing overlayed logical windows, transaction data entry and three kinds of menu systems. Features virtual windows, many data types, device independence, total screen management and more. Simple to use, yet highly sophisticated. Supports all major compilers, all models, auto installation. Over 125 functions, no royalties, source available.

GREENLEAF Software

PANEL Plus

Advanced screen manager with a screen designer, code generator and function libraries. Works with popular graphics libraries to allow data entry in graphics modes. Includes full, portable library source code.

Brand New LIFEBOAT

spII Screen Manager

Generate COBOL screen handling source code or use the powerful run-time unit! Create prototypes which include menu processing, functional I/O panels, field editing, pop-up panels and more. Create demos with your application panels. Easy-to-use panel painter with automatic borders, complete attribute control; move/copy blocks of the panel. Save overlaid screens. Extensive field editing, including range check, discrete and binary values. Secure fields for passwords, hot fields to automatically return control, required input fields.

Texus

Pascal-2

Have you reached the limits of Turbo? Upgrade to Pascal-2 without loss. Easy migration path from Turbo with compatible strings, equivalent procedures and access to Turbo graphics. Cut execution time by 200% over Turbo. Use all of DOS-accessible memory through efficient, large memory model. Speed development time with sophisticated error checking and reporting. Call Microsoft languages. Plus interactive source-level debugger, error walkback, Intel CEL87 math library, high-level profiler.

OREGON SOFTWARE

Call today for your FREE Scientific & Engineering Solutions catalog!

Describes 100 of the best software packages for solving complex equations, number crunching, analyzing data and more.

Call **1-800-847-7078**
In NY **914-332-1875**
or see your local Lifeboat Affiliated Dealer

Your source for the latest 80386 development software.

The Full-Service Source for Programming Software

LIFEBOAT

55 South Broadway
Tarrytown, NY 10591
Telex # 510-610-7602

The names of the products listed are generally the trademarks of the sources of the products.

INTERNATIONAL SALES OFFICES

Australia/New Zealand: MoS Computer Software/ Charlton Distributors Auckland (09) 766-361
Canada: Scantel Systems Toronto (416) 449-9252
Denmark: Ravenholm Copenhagen 288-7249
England: Grey Matter, Ltd. Devon 364-53499
System Science, Ltd. London (01) 248-0962
France: Compusol Paris 14 530 0737
Italy: Lifeboat Associates Italia Milan 02-464601
Japan: Lifeboat, Inc. Tokyo 03-293-4711
SATT Software Tokyo 03-295-3390
Netherlands: SCOS Automation BV Amsterdam 020-10 69 22
Spain: Micronet, S.A. Madrid 1-262-3304
Switzerland: Euro-Link Zurich 813 514
West Germany: MEMA Computer GmbH Frankfurt 069-347226
Omnitex Rheinfelden 07623/61820

FORTRAN PROGRAMMERS

LCS ANNOUNCES F77L LAHEY FORTRAN VERSION 2.20
WE JUST MADE OUR TOP RATED FORTRAN LANGUAGE SYSTEM BETTER.

"Lahey's F77L FORTRAN is the compiler of choice. It's definitely a 'Programmer's FORTRAN,' with features to aid both the casual and the professional programmer . . . F77L compiled the five files in a total of 12 minutes, which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN..."

—Editor's Choice *PC Magazine*

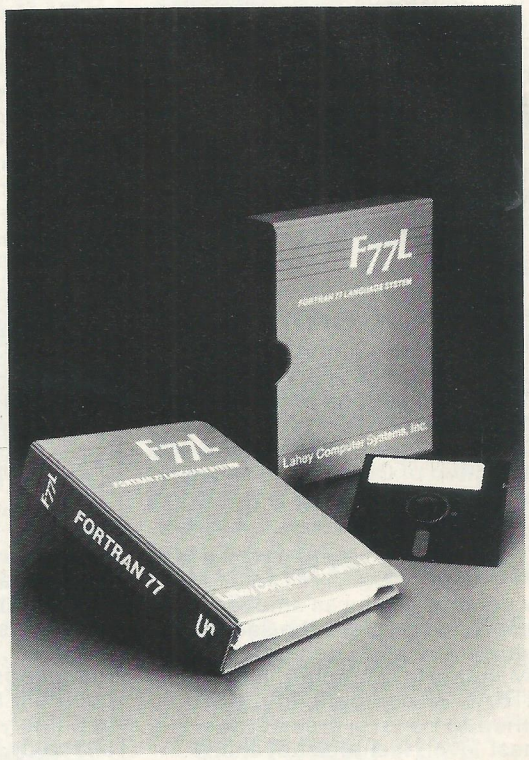
HERE ARE JUST A FEW OF THE REASONS WHY F77L IS THE COMPILER OF CHOICE:

- Full Implementation of the ANSI 77 Standard
- Fast Compilation—outruns everything on the market
- Powerful Multi-Featured Source On-Line Debugger
- Popular Extensions for easy porting of mainframe and mini computer programs (Including NAMELIST)
- Recursion—allocates local variables on the stack
- Arrays and COMMONS greater than 64K
- Clear and Precise English Diagnostics
- Long Variable Names—Up to 31 Characters
- COMPLEX*16, LOGICAL*1 and INTEGER*2
- IEEE Standard Floating Point
- Compatibility with popular third party software
- Unmatched Technical Support with an on-line bulletin board

NEW FEATURES WITH VERSION 2.20:

- Cross reference and source listings
- Allocation maps of COMMON variables and arrays
- In-line comments
- IMPLICIT NONE compiler option
- Faster Execution
- Source On-Line Debugger (SOLD) includes: Trace Execution; No Relinking required; On-screen Listing; No effect on code size

Call about our New F77L development tools:
Lahey Profiler Mathematical Functions Library Overlay Linker



It is more than just features that make F77L an outstanding product; it is the years of experience behind the software. At Lahey Computer Systems, we have been developing FORTRAN compilers since 1967 and we are committed to keeping F77L the industry leader.

When *PC Magazine* selected our version 1.35 as the Editor's Choice among PC FORTRANs, we were pleased but not completely satisfied—we knew we could improve the product. F77L Version 2.20 increases our lead over the competition. F77L's precise diagnostics, advanced debug package, helpful user screens and comprehensive manual make it a complete and easy to use high productivity tool.

When evaluating any software package, an important factor to consider is the value of your time. F77L saves you time and money the moment you start using it. Our FORTRAN Language System has the key features you need to increase productivity and get the job done. Other PC FORTRANs may be cheaper than F77L, but none are less expensive to use.

F77L—THE PROGRAMMER'S FORTRAN

Price: $477.00
System Requirements: MS-DOS or PC-DOS (2.0 or greater), 256K, math coprocessor (8087-80287)

TO ORDER OR FOR MORE INFORMATION:
702-831-2500

Lahey Computer Systems, Inc.
P.O. Box 6091, Incline Village, NV 89450
Telex: 9102401256

International Representatives:
Canada: Barry Mooney & Assoc., Tel. (902)6652941 • **England:** Grey Matter Ltd, Tel. (0364)53499 • **Switzerland:** DST Comp. Services, Tel. (022)989188
Denmark: Ravenholm Computing, Tel. (02)887249 • **Australia:** Comp. Transitions, Tel. (03)5372786 • **Japan:** Microsoftware Inc., Tel. (03)8138222

MS-DOS & MS FORTRAN are trademarks of Microsoft Corporation. Pro FORTRAN refers to IBM PC Professional FORTRAN by Ryan McFarland

CIRCLE 30 ON READER SERVICE CARD

Reliable FORTRAN with MAT

By Gerald M. Berns

In recent years, increasing attention has been given to the production of reliable programs. New programming languages have been developed specifically to improve program reliability. Some of these languages are among the largest and most complex ever devised. It is not widely known, however, that the oldest high-level language, FORTRAN, can produce programs that are just as reliable–when its use is accompanied by the application of a capable static analyzer.

A static analyzer is a tool that detects errors and problems in programs by analyzing their source code. Its diagnostic capability considerably surpasses that of a conventional compiler but does not duplicate it. The major difference in the diagnostic capabilities of the two is their scope. Although a program may consist of many modules (such as subroutines, functions, etc.), a conventional compiler analyzes them one module at a time without retaining information across modules. (The newer programming languages are designed to enable their compilers to have access to module interface data.)

A static analyzer also analyzes each module but retains information across modules. Once the individual modules have been analyzed, a static analyzer then analyzes the program as a whole. That is, a static analyzer can analyze all the source modules of a program together as the single entity that they constitute. Of the set of available software diagnostic tools, only a static analyzer can apply the powerful diagnostic tests appropriate to the analysis of a program as a whole.

The outstanding advantage of a capable static analyzer is that its diagnostic information can make new and existing software significantly more reliable. Its report enables programmers to produce reliable software without having to learn a new programming language. Its report also enables maintenance people to make existing programs more reliable without the costly reprogramming necessary to replace the language in which they are written. The availability of a static analyzer for an established language means the software community can continue to use the language as it is without any loss of reliability in the resulting programs.

FORTRAN is perhaps the most profitable language with which to use a static analyzer. On one hand, FORTRAN is simple to learn and use, is very powerful, produces small and fast modules, and is known by the majority of programmers. On the other hand, FORTRAN is notorious for the unreliable and unmaintainable programs it allows (especially large systems) and the poor programming practices it tolerates (if not encourages).

The Maintainability Analysis Tool (MAT) is a static analyzer for FORTRAN. This article describes how MAT closes several major reliability holes of FORTRAN. Examples are given that show how the combination of FORTRAN and MAT can produce programs as reliable as those produced by any language available today.

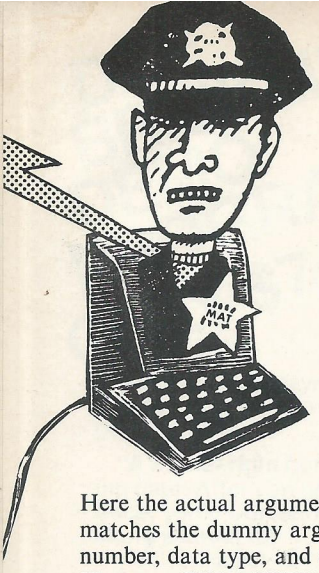
MAT runs on several mini and mainframe computers (including DEC, IBM, CDC, Cray, HP, and Prime) and is available for others that offer a full FORTRAN-77 compiler. Developed in 1984, MAT today is in use at more than 10 sites in the U.S., Canada, and elsewhere.

Argument checking

The FORTRAN-77 specification (and the dialects based on it) states that actual arguments passed in a call to another module must agree in number and type with the (dummy) arguments defined for the called module. However, because conventional FORTRAN compilers process each module of a program in isolation from (and without any knowledge of) the requirements of the other modules of the program, this rule controlling intermodule communications is very rarely enforced.

MAT diagnoses all violations of this rule. It also monitors the number, data types, and element types of arguments and checks for inappropriate actions. For example, MAT diagnoses when an actual argument is a variable and the corresponding dummy argument is an array. Furthermore, MAT diagnoses the following situation, which has caused FORTRAN programmers considerable trouble:

```
                    SUBROUTINE CUP
                      (ARG)
...                   INTEGER ARG
CALL CUP (35000)      ...
...                   ARG = ARG+5
                      ...
```



Here the actual argument (*35000*) matches the dummy argument (*ARG*) in number, data type, and element type. However, *SUBROUTINE CUP* changes the value of its argument. This action is inappropriate because the actual argument is a constant whose value must not be changed.

MAT also effectively polices perhaps the most elusive of all argument type mismatches: the use of integer constants and expressions as actual arguments in FORTRAN dialects that offer more than one integer data type (such as *INTEGER*4* and *INTEGER*2*). In the following example:

```
...                     SUBROUTINE TRAY
                          (TICKET)
CALL TRAY (5)           INTEGER*4 TICKET
...                     ...
```

the problem centers around the actual argument: What data type is 5? In some dialects 5 is *INTEGER*2*. In other dialects 5 may be *INTEGER*2* or *INTEGER*4*, depending upon the setting of a compiler switch. When 5 is *INTEGER*2*, are there FORTRAN dialects in which *SUBROUTINE TRAY* always picks up the correct value (5)? Surprisingly, the answer is yes (DEC VAX FORTRAN is one example). In most dialects, however, the answer is no, and the match-up would be in error.

In some dialects, 5 is *INTEGER*4* (IBM VS FORTRAN, for example). This results in a data type mismatch if dummy argument *TICKET* is defined as *INTEGER*2* instead of *INTEGER*4*. MAT keeps all of this straight through its FORTRAN DIALECT = name Option Statement and diagnoses all erroneous matchups.

MAT also diagnoses a subtle variation on the previous example:

```
INTEGER*2 I             SUBROUTINE
                          SAUCER (J)
...                     INTEGER*4 J
CALL SAUCER (I-3)       PRINT *,J
...                     END
```

In DEC VAX FORTRAN, *SUBROUTINE SAUCER* prints the correct value for *J* (that is, *I*−3) when *I* is greater than or equal to 3. However, when *I* is less than 3, *SUBROUTINE SAUCER* receives and prints an incorrect value. This is not a compiler error because the reference manual warns the user not to pass an actual argument of one type to a dummy argument of another type (in this case the actual argument is *INTEGER*2*). Here is a situation where a running program can sporadically fail, depending only on the value of a dynamically changing element, *I*.

Argument mismatches can plague efforts to transport a program to a computer that offers a different FORTRAN dialect, especially when the argument mismatches are caused by the change of dialect. MAT makes the transporting process more reliable by diagnosing such problems. The FORTRAN DIALECT = name Option Statement can again be used as MAT analyzes the program; in this case "name" is the name of the target computer.

Common block checking

Modules also communicate by sharing global elements. In FORTRAN, global elements reside in common blocks. A module that contains the definition of a common block has access to the elements of that common block. That is, if 50 modules in a program use elements of common block *BOOK*, then *BOOK* must be separately defined in each of the 50 modules. Therein lies the problem because traditional FORTRAN compilers examine modules in isolation and cannot detect when the same common block is defined differently in different modules. Nor can they detect when the same named element is defined as a member of two or more common blocks (unless the common blocks happen to be defined within the same module). Different definitions of the same common block can lead directly to errors; the same element name in multiple common blocks generates (and may be due to) programmer confusion and can lead to errors.

MAT detects all discrepancies in common block definitions. Using the first occurrence of a particular common block as definitive, MAT diagnoses all definitions of the same common block that deviate in any way from the first one. This diagnostic coverage includes differences in the number of elements in the common block and all differences in the definition of each element in the common block. Each element is monitored for name consistency, data type, element type, and, if the element is an array, the number of dimensions and the bounds of each dimension. Elements that are equivalenced into a common block are also thoroughly monitored, as are memory boundaries.

The protection that MAT provides FORTRAN programs extends from the detection of blatant errors to those that are quite subtle in common blocks. Suppose a system consists of 1,000 FORTRAN modules, of which 300 are intended to communicate via the elements in common block *COVER*. However, unbeknownst to the developers, 299 modules contain the correct definition:

```
INTEGER A1, A2, A3, A4
COMMON /COVER/ A1, A2, A3, A4
```

but one accidentally contains the definition:

```
INTEGER A1, A2, A3, A4
COMMON /COVEN/ A1, A2, A3, A4
```

(Note that this typo, *COVEN* instead of *COVER*, is unaffected by use of the *IMPLICIT NONE* statement offered by many dialects.) This error, which MAT has found in production systems, is legal FORTRAN and very difficult to detect by other means.

The global analysis part of the MAT report includes the "Multiply Defined Names" section, which for this example would contain two entries for each of the symbolic names *A1*, *A2*, *A3*, and *A4*. For *A1* it might say:

```
1 A1: VARIABLE, INT, ELEMENT OF
   /COVER/: 1, 3S, 4SR, 5R, 6, 7R, ...
2 A1: VARIABLE, INT, ELEMENT OF
   /COVEN/:212R
```

where *INT* means integer, *3S* means the element is set (given a value) in module 3, and *4SR* means the element is set and referenced in module 4. The module numbers are assigned by MAT, and the erroneous definition occurs in the 212th module. Similar entries would appear for *A2, A3,* and *A4*.

In the MAT report section entitled "Discrepancies in Use of Common Blocks" these diagnostics might appear:

```
***INTERFACE CHECK(94):
   UNUSED COMMON ELEMENT:
   /COVEN/A2
***INTERFACE CHECK(96):
   COMMON ELEMENT WAS REFER-
   ENCED BUT NOT SET IN ONE
   MODULE: /COVEN/A1 /COVEN/A4
***INTERFACE CHECK(98):
   COMMON ELEMENT WAS SET BUT
   NOT REFERENCED IN ONE
   MODULE: /COVEN/A3
```

Residency checking

Many FORTRAN-77 dialects offer the "include file" feature. The major benefit of this excellent extension to the FORTRAN-77 specification is that it allows entities, such as common blocks, to be specified just once: in an include file. Each module that requires access to these entities references the include file via an *Include* statement. This causes the compiler to incorporate the contents of the include file into the module.

Use of include files to define common blocks can effectively address many of the problems discussed in the previous section, but they can also introduce subtle problems of their own. Suppose, for example, that the definition of common block *COVER*:

```
INTEGER A1, A2, A3, A4
COMMON /COVER/ A1, A2, A3, A4
```

resides on an include file, and that the 300 modules requiring access to it all contain the appropriate *Include* statement. The configuration manager may rest easy in the belief that the definition of common block *COVER* is now under full control, and that each module accessing common block *COVER* will always have the identical definition of it. However, many MAT analyses indicate that such is not the case.

What MAT has found in analyzing very large systems (of more than 150 modules) is a breakdown in configuration management. Systems of this size are prone to contain at least one module in which a common block, such as *COVER,* is hard-coded. Perhaps the module predates the migration of *COVER* to the include file; perhaps one programmer never got the word. Whatever the reason, MAT has found this situation when analyzing a number of large production systems.

Errors arise from this "split residency" (include-file-resident and hard-coded) condition as the system evolves and common block *COVER* changes. Although the configuration manager carefully controls the master copy resident on the include file, he or she is not aware of the hard-coded instance. The result is that the hard-coded definition of common block *COVER* does not change in concert with its other instances, which can introduce errors that are very difficult to locate.

MAT plugs this hole in its global analysis section, "Residency Discrepancies":

```
***INTERFACE CHECK(67):
   COMMON BLOCKS BOTH HARD-
   CODED INTO MODULES AND
   INCLUDE-FILE-RESIDENT: COVER
```

which is followed by the later reminder:

```
***INTERFACE CHECK(258):
   THIS MODULE SET CONTAINS BOTH
   HARD-CODED AND INCLUDE-FILE-
   RESIDENT COMMON BLOCKS
```

Furthermore, as previously described, MAT diagnoses all the differences that may exist between the various versions of common block *COVER*.

MAT applies similar diagnostic protection to named constants (parameters) that may be both include-file-resident and hard-coded.

Referenced but not set

More than with any other language, programs written in FORTRAN are plagued by typographical errors. Typos of all sorts (missing, extra, wrong, and transposed characters) frequently have legal results. For example, consider this code from a recently analyzed production system:

```
REWIND THE INPUT UNIT
REWIND (UNIT=10)
```

The comment indicator (C or * in position 1) was inadvertently omitted from the first line. This results in the spontaneous definition of a new element named *THEINPUTUNIT* (legal in the dialect used). Because the new element has no explicit value, this dialect assigns it the value zero. Zero, however, is a valid I/O unit number. So what was intended to be an informative comment instead winds up being a perfectly legal instruction to rewind I/O unit zero!

MAT detects and diagnoses all spontaneously defined elements, completely filling this reliability hole of FORTRAN. MAT also diagnoses all instances where an element that never receives an explicit value is referenced. This protection applies to global as well as local elements, and all equivalencing is taken into account.

Most FORTRAN dialects do not support circular (recursive) calling; a few only support it for subroutines. The problem is that conventional compilers and linkers do not detect circular calling when it is illegal. When a program consists of many modules, occasionally unintended and unsupported circular calling creeps in. This can lead to program failures of various kinds, including endless looping and resource exhaustion followed by an abort.

MAT has found instances of circular calling even in production programs. Interestingly, they are usually found in the portions of large systems that perform error handling! There apparently is a tendency among programmers to invoke

error processing logic, even within the error processing logic itself.

In the "Call Tree" section of its report, MAT detects and diagnoses all forms of circular calling. MAT's call tree is presented by task in hierarchical order. The Call Tree displays not only all the modules that each module calls but also all the modules that call it.

Program-wide mapping

In programs that consist of many modules, errors sometimes occur because either the developer or the maintainer does not know all the modules in which particular global elements are set or referenced. The problem arises when corrections or enhancements require global elements to be altered. Such changes usually require that all instances of the global elements be reviewed and the appropriate ones revised. However, if no one knows all the modules in which the elements appear, the revisions may not be made to all appropriate instances. This problem afflicts all languages, not just FORTRAN.

MAT solves this problem by providing a cross-reference facility. The cross-reference table contains an alphabetical listing of every symbolic name used in the entire program by element type (such as variable, array, and common block). The entry for each element contains its name, element type, the module numbers in which it appears, and what is done (set, referenced, used as an actual argument) with the element in each module. Type-dependent information is also provided. For example, a global array's data type, dimensions, dimensional bounds, and the name of its common block are given; for a common block, all of its elements (including a separate grouping of those equivalenced into the common block) and its residency are reported.

The cross-reference table identifies all the modules in which each element occurs. In addition, for each module MAT documents all of the dummy arguments, include files, common blocks, subroutines, functions, and I/O units (plus all I/O actions performed on them) present in the module.

MAT + FORTRAN

I have attempted to show how one static analyzer, MAT, closes major reliability holes of FORTRAN. Rather than design a language specifically to produce more reliable programs and accept the cost of a new and complex language with few practitioners and large and unproven compilers, the FORTRAN/MAT approach offers a viable alternative. It marries the well-known virtues of one of the most widely used languages in the world, with its many excellent and time-tested compilers, to a separate tool specifically designed to detect the reliability problems to which the language is vulnerable. The result is efficient programs that are as reliable as those produced by any other language.

This distribution of functions between FORTRAN and MAT allocates to each what it does best and imposes no burden of inefficiency on the resultant executable program. To the best of our knowledge, MAT has never examined a production FORTRAN program of any significant size without finding errors. Analyzing large production systems, MAT usually finds 10-50 errors and many more questionable areas. After MAT's diagnostics and other documentation are used to correct the problems, the modules are recompiled by FORTRAN compilers well known for their small, fast products. That is, once MAT's diagnostic function is performed and the program is corrected, no performance cost associated with that diagnostic function is imposed on the program.

MAT runs approximately as fast as the FORTRAN compiler available on the same machine. Its capacity is quite large: in a single run MAT has analyzed systems that contain more than 1,600 modules, 100 tasks, and 12,000 symbolic names. Its ability to handle extended FORTRAN dialects is also large. For example, MAT supports seven representations of hexadecimal constants, eight ways to express octal constants, and 10 forms of *Include* statements.

We intend to continue expanding the number of FORTRAN forms that MAT supports and the machines on which it is available. We believe that MAT's diagnostic and documentation capabilities should be available to everyone who needs to produce reliable programs. Thorough static analysis is just one of the elements necessary to achieve reliable

How to obtain and run MAT

MAT runs on these mainframes and minicomputers: DEC VAX and Microvax, IBM, Prime, Data General, CDC, Hewlett-Packard 1000, HP-9000, and Cray.

To run on any computer, MAT requires that the target computer have available a full FORTRAN-77 compiler that supports the following:

- Bit-oriented functions: inclusive *or,* exclusive *or, and, not* (1's complement), and logical shifts (left and right)
- A4 and A2 edit descriptors used by integer I/O list items
- Extended *If, Else If,* and *Else* blocks (these allow a branch out of a block to be followed by a branch back into the same block).

The largest arrays in MAT are dimensioned for 262,140 bytes (in a common block) and 100,000 bytes (local). These can be decreased if necessary.

MAT does not presently run on a PC, but the author would be happy to talk to PC users interested in MAT.

For further information, contact the author at SAIC Corp., 1213 Jefferson Davis Hwy., Suite 1500, Arlington, Va. 22202, (703) 979-5910.

programs, of course. But it is an important element, one whose benefit to cost ratio is quite high.

For example, a few years ago an error in a NASA program achieved a place in the history of computer science. Instead of entering a *Do* statement like:

```
DO 100 I = 1, 3
```

the programmer inadvertently substituted a period for the comma, resulting in:

```
DO 100 I = 1. 3
```

The FORTRAN compiler interpreted this line as a perfectly correct, albeit unintended, assignment statement that sets variable *DO100I* to 1.3. This error was not found during testing and is said to have cost NASA the loss of a Venus probe. Had MAT been available then, it might have saved NASA from this misfortune. MAT detects this error (even if an element named *DO100I* is explicitly defined) and announces it clearly:

```
***FORTRAN ERROR(196):
   DO STATEMENT HAS INCORRECT
   NUMBER OF ITERATION CONTROL
   EXPRESSIONS AT LINE ____
```

A static analyzer can help to prevent costly errors such as this one from occurring in the field. The application of a capable static analyzer to FORTRAN programs in the development and maintenance phases has proved to be a powerful and cost-effective way to increase the reliability of FORTRAN programs.

Gerald Berns is director of software products at SAIC Corp. He is the developer of the MAT, PAT (Path Analysis Tool), and REFLECS tools for FORTRAN users. He has also worked extensively in the areas of error management, resource management, real-time systems, test drivers, and file management systems.

Artwork: Anne Doering

UNIX/C WINDOW DEVELOPMENT
COMPATIBILITY with **CURSES** for MS-DOS.

Buy the window development tool that Unix/C experts demand!

Let the speed, versatility and ease of using Aspen Scientific's Curses for MS-DOS improve your products' user interface—all this plus Unix/Xenix source code compatibility for you, the serious developer.

Our Curses can do any job you require from a window tool. It will save you time, money, and your reputation. Our Curses blazes on IBM PC's and compatibles, but is versatile enough to work under MS-Windows and on any personal computer running MS-DOS 2.0 and later.

Look at a few of the applications our Curses is benefiting:

- Voice Recognition
- Telephone Switching
- Environmental
- Aerospace
- Financial
- Consulting
- Robotics
- Engineering
- Hobbyist

Compilers: Microsoft, Lattice, Aztec, DeSmet
Trademarks: MS-DOS, MS-Windows (Microsoft); Unix (AT&T Bell Labs); Lattice (Lattice, Inc.); Aztec (Manx Software Systems); DeSmet (DeSmet Software).

Complete curses tool kit: $119.
Source code available for: $289.

ASPEN SCIENTIFIC
P.O. BOX 72 WHEAT RIDGE,
COLORADO 80034-0072
(303)423-8088

CIRCLE 31 ON READER SERVICE CARD

The *cito* Environment
An Interactive Program Development Environment

The *cito* environment is a revolutionary new program development tool which will greatly increase the productivity of any C programmer. *Cito* allows you to create, run and debug procedures in an ***interactive*** environment. When you make a change to your program only the changed module is reloaded into the environment. All the existing code and global data remain unchanged. ***No longer does a small change require relinking an entire program!*** The *cito* environment uses your favorite C compiler, editor and other programming aids, controlled from a powerful macro interpreter. Finally, the speed, reliability, and portability of compiled C with the convenience of an interpreter!

- Fast reloading of individual modules.
- Command line calling of individual routines.
- Complete error handling and recovery.
- No decrease in run-time efficiency.
- Powerful, general purpose macro interpreter which generates compiled code.
- Fast multitasking within the *cito* environment.
- Easy to learn — interactive tutorial included. Provides a great environment for learning C.
- Available soon — source code debugging of C routines.

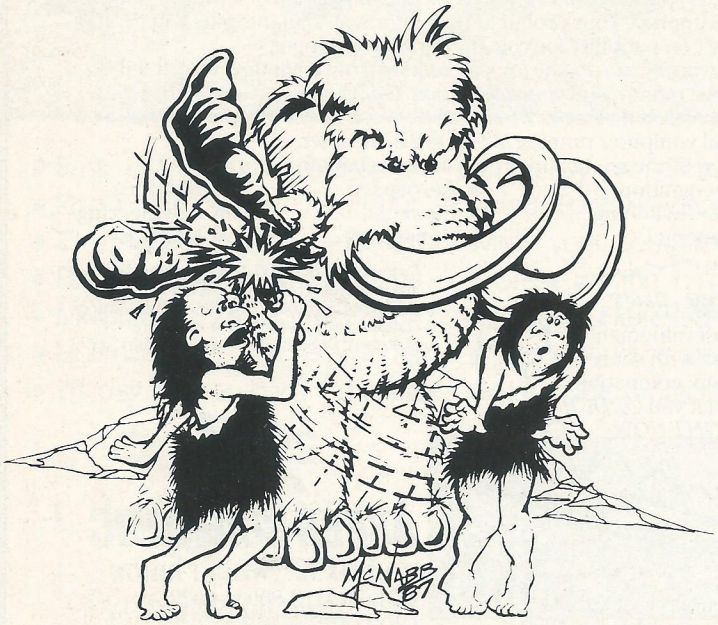
Cito makes it easier than ever to write large or small application programs with unparalleled speed! *Cito* is now available for Xenix* 286, and available soon for a wide variety of hardware. The *cito* environment for Xenix 286 is priced at $229. For more information, contact:

Fillmore Systems, Inc. (612) 831-6984
7200 York Ave. So., Suite 301, Edina, Minnesota 55435

*Xenix is a trademark of Microsoft Corp.

CIRCLE 32 ON READER SERVICE CARD

MAMMOTH PROJECTS OFTEN FAIL WITHOUT THE RIGHT TOOLS.



WITHOUT THE PROPER TOOLS ANY COBOL APPLICATION CAN BE A MAMMOTH PROJECT

And writing your own code to make your screens and menus look good can add weeks or even months to the development of any COBOL program.

Source Code Plus Productivity.
COBOL spII gives you the best of both worlds... fast, easy generation of COBOL screen handling source code and highly sophisticated panels.

Runtime or Source Code . . . Your Choice.
If you want a smaller COBOL program, you can use *COBOL spII's* runtime unit which resides outside your source program. With *COBOL spII*, you make the choice . . . we don't try to make it for you.

Real Prototypes . . . Not Slide Shows.
COBOL spII gives you a powerful dialogue facility to help you create interactive demos and tutorials using your application panels. And you don't write a single line of COBOL source code for the dialogue.

We're not talking about a puny on-line slide show, *COBOL spII* prototypes give you real menu processing, functional I/O panels, extensive field editing, pop-up panels and more. This makes it easy to see how your program will really look.

What's more, you can create continuously running demos or tutorials complete with beep control so that your demos will sound as good as they look!

Panel Painting . . . The Way It Should Be.
COBOL spII gives you a powerful panel painter which automatically sets panel attributes, generates automatic borders and allows you to move or copy blocks of the panel within or across panels.

Complete Field Editing and Validation.
Practically all of your field editing can be done while you paint your panels. *COBOL spII* performs range and discrete value checking as well as binary value checking.

Field Types for Every Purpose.
In addition, *COBOL spII* gives you a wealth of field types and formats to perform the functions you need to accomplish in your program. Field types include secure fields for passwords, three different date field formats, protected fields for creative menu design and of course the standard input, output and numeric fields. Hot fields to automatically return control to your program and required input fields are just a couple of the many field formats offered.

Changing Compilers? . . . No Problem.
One *COBOL spII* package supports the four major PC COBOL compilers. If you have RM COBOL, Realia COBOL, Microsoft COBOL or RM COBOL 8X, *COBOL spII* can help you generate applications in a fraction of the time that it takes you now.

Special Offer . . . Ends August 31st, 1987!
Better give us a call! For a limited time we're offering *COBOL spII* for only $345.00. After August 31st, 1987, the normal retail price of $395.00 will go into effect. Save even more money and contact your favorite program tools distributor today!

Still Skeptical? . . . Call for Our Free Demo.
Seeing is believing. Give us a call and we'll send you a free demo disk. Then you can see the power, flexibility and ease-of-use of *COBOL spII*.

Not Happy? . . . 30 Day Money Back Guarantee!
We're confident that you'll find *COBOL spII* to be as good as we say it is. That's why we'll give you a 100% money back guarantee if you're not completely satisfied. Just return the package in good condition within 30 days of purchase. . . . no questions asked.

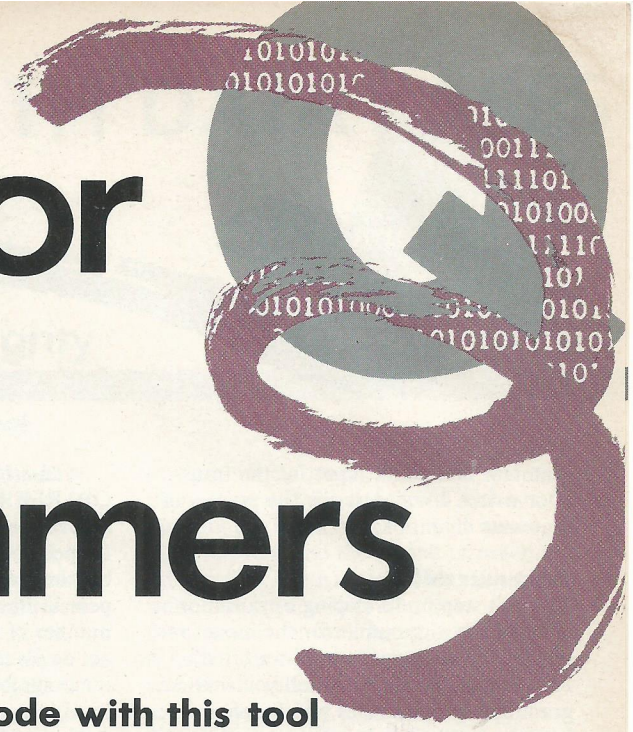
FLEXUS

COBOL spII . . .
the missing link
to COBOL productivity!

FLEXUS INTERNATIONAL CORPORATION ■ P.O. BOX 9119 ■ MORRISTOWN, NJ 07960 ■ (201) 895-4724

CIRCLE 33 ON READER SERVICE CARD

A Translator for Lazy Programmers



Convert Pascal statements into assembly code with this tool

By Todd Jonathan Egan

I dread the prospect of having to write any computer program in assembly language. Like most programmers, I prefer the logical structure and readable English syntax of a high-level language like C or Pascal. Unfortunately, circumstances beyond my control sometimes dictate that I write a segment of code or, God forbid, an entire program in assembly language.

To overcome my aversion to assembly language programming, I have been developing a programming environment called LASM (lazy assembly) that allows me to convert Pascal statements into 8086/8088 assembly language code within my editor. Some of the greatest ideas are born out of laziness, and this Pascal-to-assembly language translator is one of them.

LASM is a combination editor and code generator. I can use it to edit my assembly code and translate high-level Pascal statements into 8086/8088 assembly language. The 8086/8088 code generated by the program may then be selectively placed into the file being edited. The samples of LASM-generated code included in this article are very simple, and some of the code generated is not fully optimized.

The reason for this apparent oversight is that I am not trying to give an in-depth analysis of Pascal-to-assembly translation. Rather, I want to present an overview of the techniques that can be used to implement language-to-language translators since these techniques may also be used to write a wide variety of other powerful applications.

When I first started to develop LASM, I knew I would have to design an editor. The Borland International Turbo Editor Toolbox made this task relatively painless. The challenging part of the program was the development of a module that would convert my input Pascal statements into 8086/8088 assembly code. At first glance, this seemed an insurmountable task.

Since LASM is essentially a compiler (LASM translates high-level Pascal statements into 8086/8088 assembly code rather than binary 0s and 1s), I looked at two programs: yacc (Yet Another Compiler Compiler) and lex. (For more about lex, see "Learning about LEX," *COMPUTER LANGUAGE,* Feb. 1987, pp. 51-56.)

yacc and lex were written to aid in the development of compilers in the UNIX environment and are available as part of the UNIX system. I do not have access to a UNIX system, and I had heard that the pirated versions available on the IBM PC were unreliable. Fortunately, I was able to find a little-known programming tool called a translator writer that facilitated the task of writing a program to convert Pascal statements into assembly code.

Translator writers

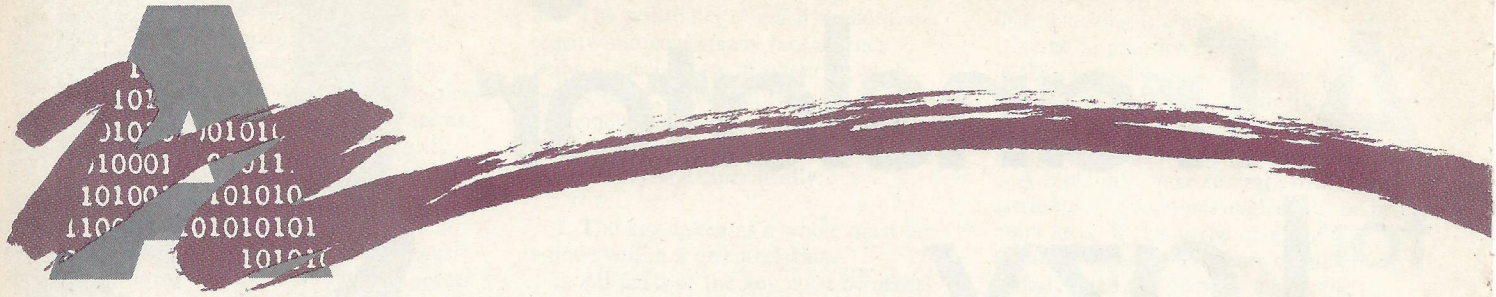
I believe that translator writers potentially have a much wider application in program development than many programmers realize. But what's a translator writer, and more specifically, what's a translator?

Since a computer can only understand a language of 0s and 1s, languages consisting of anything but 0s and 1s must be translated into the 0s and 1s a computer can understand. Compilers that translate high-level computer language statements into the 0s and 1s a computer can understand are the most common application of this technology.

Although translator writers are used primarily in the development of compilers and interpreters, they are also very useful in the programming of any language-intensive project, including high-level, language-to-language translators; syntax checkers; and user interfaces. Since the translator writer seemed ideally suited to my needs, I purchased from QCAD Systems a translator writer called QPARSER+, which is available on the IBM PC. (QPARSER+ is described by its developer, William Barrett, in "Source Code Generators," *COMPUTER LANGUAGE,* Jan. 1987, pp. 59-70).

LASM translates Pascal into assembly code, but as I explain how LASM was developed, you should be able to see how these same techniques could be used to develop other language translator programs (Pascal-to-C, BASIC-to-Pascal, dBASE III-to-C, etc.). A user interface is another good application for the translator writer since an interface translates a user's commands into actions.

The translator writer will automatically generate a syntax-checker from the information in a grammar file. Since this



grammar is the first input for the translator writer, I will describe the process of grammar definition.

Grammar definition

The first step in developing a translator is to specify a grammar for the input language (the language being translated). Any English teacher will tell you that a grammar specifies rules which define how the words in a language may be combined into sentences, but how do we write rules that define how the words in a computer language like Pascal may be used in statements?

The process of grammar definition is relatively simple to do but difficult to explain, so let's start with a sample rule from my Pascal grammar:

```
statement —> IF bool
   THEN statement ELSE statement
```

This rule says that a *statement* may consist of the word *IF* followed by a *bool* (short for Boolean), a *THEN*, another *statement*, an *ELSE*, and yet another *statement*. The elements of the rule in uppercase letters are the actual words constituting the syntax of the language. The words in lowercase letters may be considered variables or placeholders that are defined elsewhere in the grammar.

I have defined the structure for one *statement* from this language, but I am free to define others as well:

```
statement —> WHILE bool DO
   statement
```

Here, I am expressing the notion that it is legal for this *statement* to appear in place of the *statement* variable in my other rule:

```
IF bool THEN WHILE bool DO statement
   ELSE WHILE bool DO statement
```

The grammar rule that defines the variable *bool* is:

```
bool —> <identifier> boolop expr
```

<identifier> is a special QPARSER+ variable that stands for any variable name. Since it is impossible to specify every possible string, real number, integer, or identifier that might appear in Pascal, QPARSER+ provides a number of these special symbols, which act as placeholders for larger sets. The other symbols include *<integer>* for fixed integers, *<string>* for quoted strings, and *<real>* for real numbers. I defined the variable *boolop* as:

```
boolop —> >=
      —> <=
      —> =
      —> >
      —> <
```

Given the grammar I have defined so far, the rule:

```
bool—> <identifier> boolop expr
```

specifies that a *bool* may consist of any identifier followed by a Boolean symbol, followed by the variable *expr*, which has yet to be defined. Since I needed a grammar rule to define a Boolean operation like *foobar > 6*, *expr* could be defined as follows:

```
expr —> <integer>
```

using the special QPARSER+ variable name for "any integer."

I also wanted my language to support Pascal syntax like:

```
foobar > (6+6)*7/10
```

so I used a further set of rules to define some simple mathematical functions:

```
expr —> expr+term
     —> expr-term
     —> term
```

where *term* is defined as:

```
term —> term*primary
     —> term/primary
     —> primary
```

and primary is defined as:

```
primary —> ( expr )
        —> <integer>
        —> <identifier>
```

The ability to express lexical precedence within a grammar is another advantage to using the translator writer. I could instead have defined a set of mathematical functions with the rules:

```
term —> term*primary
     —> term/primary
     —> term+primary
     —> term-primary
     —> primary
```

but these rules do not express mathematical hierarchy (that is, multiplication and division should be evaluated before addition and subtraction). The hierarchical structure of the rules I used dictates that the parser generated by this translator writer will have to evaluate the *term* in the rule:

```
expr —> expr+term
```

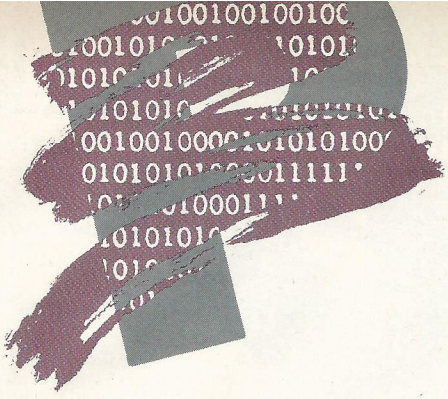
before it can evaluate the addition of *expr* and *term*. Given the input phrase *5+4*3*, the parsing program generated by the translator will automatically evaluate the term *(4*3)* before evaluating the addition *(5+12)*.

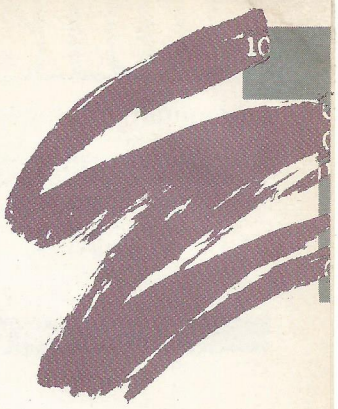
These rules define any simple mathematical equation from the one variable *expr*. For example, the equation 5+5 is defined by the rule:

```
expr —> expr+term
```

because *expr —> term, term —> primary,* and *primary —> <integer>*. The equation *foobar+foobar* is also defined by these rules since *primary —> <identifier>*.

Since the grammar variable *expr* may expand into any simple mathematical expression, I can use it in a number of other Pascal grammar rules as well. For example, the Pascal assignment statement may be defined as:





statement —> <identifier> := expr

This rule supports Pascal statements like:

foobar := (6+6)*7/10

I could also use the variable *expr* in a production to define a *FOR-NEXT* loop:

statement —> FOR <identifier> :=
expr TO expr DO statement

to define a Pascal statement like:

for foobar:= 6 TO 4*(5+5) do ....

For the sake of simplicity, I will only support the use of integer values as the index in my *FOR-NEXT* loop:

statement —> FOR <identifier> :=
<integer> TO <integer> DO
statement

The Pascal subset grammar I have defined so far appears in Listing 1. The grammatical derivation of a sample assignment statement defined by my grammar (*foobar:=6+6*7/3*) is shown in Listing 2.

Now that I've provided a brief overview of grammar definition, let's look at how the very abstract process of defining a grammar can save hours of programming effort.

Translation the easy way

Grammar definition for a complex language like Pascal is not only challenging but also rewarding. The grammar processing program of the translator writer can take the information in the grammar and produce source code for a custom language parser, symbol table functions, and tree-building features that greatly simplify the development of translator programs.

Running a grammar through the processing program of the translator writer will generate source code for a complete language parsing program customized for the language defined by that grammar. QPARSER+ can generate parsers in either C or Pascal source form.

The generated parser is ready to accept and check any of the statements defined by my grammar. The parser checks the syntax of the input statements, so if I input the statement:

```
3 := 1;
^Error: Found <integer>, expecting
   <identifier>
```

the program will report a syntax error since my grammar does not specify that it is legal to assign a value to an integer. The parser also specifies the offending character and an error message.

Syntax checking is only the tip of the iceberg of features provided by a translator writer, though. The generated parser also provides facilities for easy evaluation and translation of the input Pascal language into 8086/8088 assembly code.

For example, if I input the statement:

foobar:= 5*8+9*10

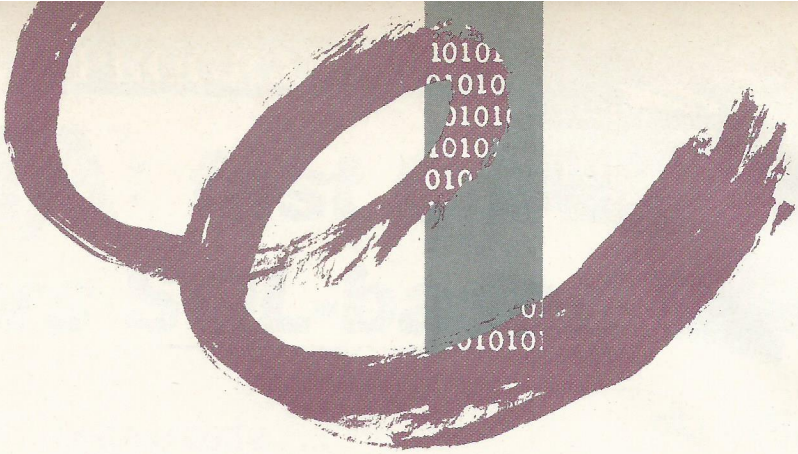
the parsing program generated by the translator writer will automatically gen-

```
statement -> <identifier> := expr
          -> IF bool THEN statement ELSE statement
          ->  FOR  <identifier>  :=  <integer>  TO  <integer> DO statement

bool -> primary boolop expr

boolop -> >=
          -> <=
          -> =
          -> >
          -> <

expr -> expr + term
          -> expr - term
          -> term

term -> term * primary
          -> term / primary
          -> primary

primary -> ( expr )
          -> <integer>
          -> <string>
          -> <identifier>
```

Listing 1.

```
              foobar:=6 + 6 * 7 / 3
statement -> <identifier> := expr
          -> <identifier> := expr + term
          -> <identifier> := expr + term / primary
          -> <identifier> := expr + term * primary / primary
          -> <identifier> := term + term * primary / primary
          -> <identifier> :=  primary  +  primary  * primary / primary
          -> <identifier> := <integer> + <integer> * <integer> / <integer>
```

Listing 2.



erate a tree structure that contains each of the key elements from the statement (Figure 1).

(QPARSER+ includes a wide variety of symbol table functions, which I use to store information about identifiers, such as *foobar*. The QPARSER+ symbol table uses standard hashing routines to store the names and types of identifiers such as *FOOBAR* as well as Pascal keywords, such as *IF, THEN,* and *ELSE*. Since compiler symbol table strategy is a subject that could take up an entire article, I will not discuss it here.)

Tree structure with key elements

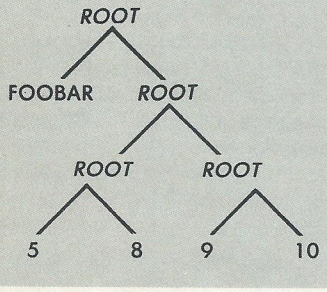


Figure 1.

Tree from new parser

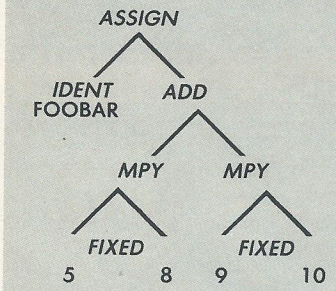


Figure 2.

Tree for FOOBAR:=6

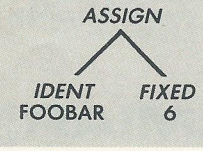


Figure 3.

Notice that the mathematical hierarchy is expressed in the structure of the syntax tree—the leaves that contain the elements to be multiplied are lower than the leaves containing the elements to be added.

You may notice that the Pascal language symbols like *, −, +, etc., do not appear in the tree. Without these language symbols, I have no way of knowing how the numbers in this tree are related to one another in the Pascal input. The 5, which appears at the lower left of the tree, might have been multiplied by the 8 or the two numbers might have been added.

To overcome this problem, I can label the QPARSER+ grammar rules with special flags that will later appear on the leaves of the trees. I add these special flags to the rules associated with the tree structure in Figure 1:

```
statement —> <identifier> :=
   expr #ASSIGN
```

and:

```
expr —> expr+term #ADD
term —> term*primary #MPY
```

In QPARSER+, these flags are called semantics types. Each of the semantics types will eventually appear as a label on the appropriate tree roots. For example, if I generate a new parsing program based on these changes to the grammar, then the tree constructed for the same input phrase by this new parser appears (Figure 2).

Each leaf of the tree in Figure 2 is labeled with a semantics type, and certain leaves contain the key values from the input statement. (Semantics types are shown in italics and the values in standard font.)

Notice that the special QPARSER+ symbols automatically end up with special labels of their own—*ident* for <*identifier*> and *fixed* for <*integer*>.

Having these key elements from the input phrase automatically placed into a tree structure allows me to write the heart of the entire Pascal-to-assembly translator within one procedure using classic recursive top-down tree-walking algorithms.

Generating a translation

The specific code to implement the translation of each subset Pascal statement into assembly language depends on the tree structure that has been generated by the translator writer. For a statement like *foobar:=6*, the tree constructed by the translator writer appears as in Figure 3.

In evaluating this Pascal assignment statement, I need to generate the assembly code to move the element on the right side of the tree into the element on the left side. The complexity of the QPARSER+ semantics data structure prevents me from listing it here, but there are some basics we need to know to implement a translation of this sort:

1. The variable that points to the root of the tree being evaluated will be passed into the evaluation procedure under the name ROOT:

```
PROCEDURE TREE_EVAL
  (ROOT:semrecp);
```

(Type *semrecp* is short for semantics record pointer.) For example, the root of the tree constructed for the assignment statement *foobar:=6* (Figure 3) has a semantics type of *ASSIGN* as a result of the tag I added to the grammar rule. Thus:

```
root^.semt = ASSIGN
```

2. The QPARSER+ data structure allows you to refer to the pointer to the first child or leaf of any root as *root^.one*. For the tree in Figure 3, *root^.one* is a pointer to the structure that contains the variable *foobar*.

The pointer to the second leaf of the tree (6) may be referred to as *root^.two*. Additional leaves may be accessed as *root^.three, root^.four,* etc. The semantics type of each leaf may be accessed as well. For example, *root^.one^.semt = ident*.

3. QPARSER+ allows the string values of each of the tree leaves to be ac-

cessed with a special function called *NAMEOF*. The statement:

```
nameof(root^.one)
```

for the root of the *ASSIGN* tree in Figure 3 returns the string *foobar*.

I use this basic information about the QPARSER+ data structure to construct a rudimentary tree evaluator. The code to generate an assembly language translation for the *ASSIGN* tree in Figure 3 is shown in Listing 3. When it is passed the root of the *ASSIGN* tree, this evaluator generates the assembly code:

```
MOVW FOOBAR, 6
```

since *NAMEOF(ONE)=FOOBAR* and *NAMEOF(TWO)=6*.

(All of the assembly code produced by the procedures listed in this article is designed to run on the CHASM assembler, which has a slightly different syntax than Microsoft's assembler.)

It is interesting to note that I could just as easily have generated a translation from Pascal into C with the statement:

```
ASSIGN: begin
   write(nameof(one),'=',nameof(two));
end;
{In C, the assignment operator is =,
   not :=}
```

Neither one of these translations seems very impressive at this point, but as we continue you will see the flexibility of the tree-based translation methods afforded by the translator writer.

Expanding the tree evaluator

The LASM *TREE_EVAL* procedure will not produce the proper assembly code if the right side of the tree does not contain a *fixed* value. For example, the expression *foobar:=foobar2* or an arithmetic expression like *foobar:= 5+5* will produce the invalid 8086/8088 assembly statements *MOVW FOOBAR, FOOBAR2* or *MOVW FOOBAR, 5+5*, respectively.

To provide for these statements, I must generate assembly code to move the values of *FOOBAR2* or 5+5 into the identifier on the left hand of the assignment via an intermediate register like *AX*.

I can test to see if the semantics type of the second leaf of the *ASSIGN* tree is an integer value (such as 5) with the statement:

```
if two^.semt = fixed then ....
```

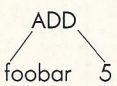
If the second leaf of the tree does not contain a *fixed* value, then the program must generate additional code to place the value contained in the second leaf of the tree into the *AX* register. A recursive call to the *TREE_EVAL* procedure will accomplish this (Listing 4).

Here, I have added another tag to my *CASE* statement to generate code to place the value of an identifier into the *AX* register. The statement *NAMEOF(ROOT)* listed under the *IDENT CASE* tag will return the string *foobar2* for the input statement *foobar:=foobar2*. Thus the *CASE* tag for the semantics type *IDENT* will generate the assembly code *MOV AX, FOOBAR2*, which places the contents of *FOOBAR2* into the *AX* register where the code for the *ASSIGN CASE* tag can expect it (Listing 5).

If the expression on the right side of an assignment is an arithmetic expression (for example, *foobar:=5+5*), then we want to evaluate the expression and place its result in the *AX* register, as with the value of an identifier. To do this, I will write a separate *CASE* tag to evaluate an *ADD* tree:

```
ADD : begin
        tree_eval(one);
        write('ADD AX, ', nameof(left));
      end;
```

This code works well for a statement like *foobar+5*, which has the tree structure:

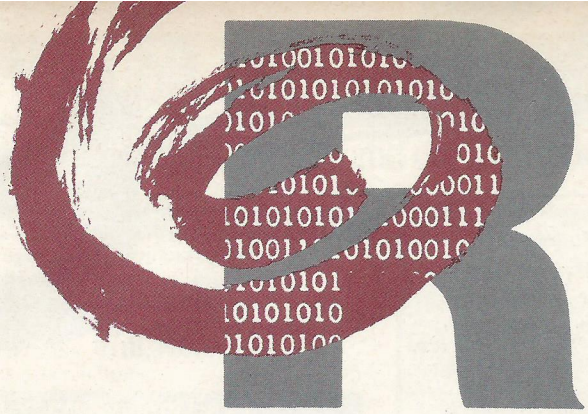



```
PROCEDURE TREE_EVAL(ROOT:semrecp);
begin
  with root^ do
   case semt of
     ASSIGN : begin
                write('MOVW',nameof(one),',',nameof(two));
              end;
  end;
 end;
```

Listing 3.

```
PROCEDURE TREE_EVAL(ROOT:semrecp);
begin
 with root^ do
    case semt of
  ASSIGN : begin
             if two^.semt = fixed then
               write('MOVW ',nameof(one),', ',nameof(two))
             else begin
              tree_eval(two);
             end;
           end;

  IDENT : write ('MOV AX, ', nameof(root));
     {.....code omitted.....}
```

Listing 4.



because the call to *TREE_EVAL* for the first leaf will generate code to place the contents of *FOOBAR* into the *AX* register. (The semantics type of *FOOBAR* is *IDENT*, which I have already written a *CASE* tag for.)

A statement such as *5+5* requires me to add another entry to my *CASE* statement to generate code to place integers into the intermediate *AX* register as I did with identifiers:

```
FIXED: write('MOV AX, ', nameof(root));
```

Now *TREE_EVAL* will produce the code:

```
MOV AX, 5
ADD AX, 5
```

for the expression *5+5*. The statement *foobar:= 5+5* will generate the code:

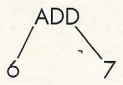
```
MOV AX, 5
ADD AX, 5
MOV FOOBAR, AX
```

To support code generation for a statement like *5+(6+7)*, I must write more elaborate code for the *ADD* tag of this *CASE* statement (Listing 6).

Function *IS_SIMPLE* tests the second leaf of the tree root to determine if it is a simple expression like 5 or *foobar* as opposed to a more complex structure like *6+7*.

If the second leaf of the tree is not a simple value, then we must make a recursive call to *TREE_EVAL*, passing the second leaf of the tree as the new root.

For the input statement *5+(6+7)*, the second leaf contains this tree structure:



As I demonstrated earlier, a recursive call to the *TREE_EVAL* procedure for a simple addition tree structure will cause the tree evaluator to generate code to place the value of *6+7* into the *AX* register.

The code generated by the evaluator for the second leaf of the tree is:

```
MOVW AX, 6
ADDW AX, 7
```

After the program returns from the first recursive call to *TREE_EVAL*, further recursive evaluation needs to be done for the first leaf of the tree, which in this case contains the number 5.

The semantics type of 5 is *fixed*, so a second recursive call to *TREE_EVAL* will generate assembly code to place the value of 5 into the *AX* register, where the value of the second leaf containing the expression *6+7* resides. I generate a *PUSH AX* instruction to save the value returned from the second leaf of the *ADD* tree onto the stack. The *AX* register is then free to store the value returned from the second recursive call to *TREE_EVAL*.

Popping the value of *6+7* off of the stack and into the *DX* register and adding it to *AX*, which contains the value 5, will complete code generation for the equation *5+(6+7)*. The full assembly code generated by this equation is:

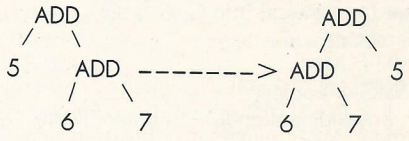
```
MOVW AX, 6
ADDW AX, 7
PUSH AX
MOVW AX, 5
POP CX
ADD AX, CX
```

This code obviously is not optimized since the instructions:

```
MOVW AX, 6
ADDW AX, 7
ADDW AX, 5
```

would have produced the same results. The code needed to optimize a tree structure is fairly easy to write, but I don't have the space here to describe it at length.

To generate this optimized code, I must switch the location of the simple tree leaf (*5*) with the complex tree leaf *(6+7)*:



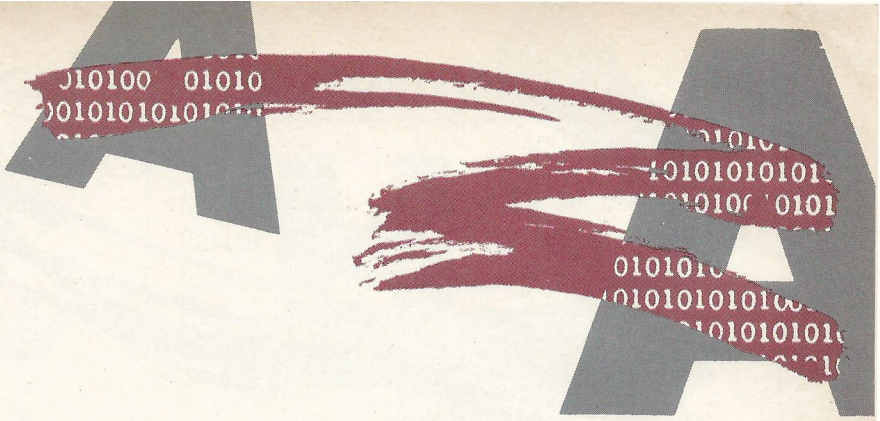
The tree optimizer will rearrange the tree so that the simple value always appears on the left-hand side of the tree. If you trace through the *TREE_EVAL* procedure for the optimized tree on the right, you will find that the procedure produces optimized assembly code for

```
{.........code omitted.........}
  ASSIGN : begin
             if two^.semt = fixed then
               write('MOVW ',nameof(one),',',nameof(two))
             else begin
              tree_eval(two);
              write('MOVW ', nameof(two), ', AX');'
             end;
           end;

  IDENT : write ('MOV AX, ', nameof(root));
```

Listing 5.

```
ADD: begin
       if is_simple(two) then begin
         tree_eval(one);
         write('ADD AX, ',nameof(two));
       end
       else begin
         tree_eval(two);
         write(PUSH AX);
         tree_eval(one);
         write(POP DX);
         write('ADD AX, DX');
       end
     end;
```

Listing 6.



that optimized tree structure.

The code to evaluate an *MPY* tree follows the same basic logic as for the *ADD*:

```
MPY: begin
       tree_eval(two);
       write('PUSH AX');
       tree_eval(one);
       write('POP CX');
       write('IMULW CX');
     end;
```

Let's look at the code generated for an input statement like *(5+5)*6*, which has the tree in Figure 4.

The statement *tree_eval(two);* will generate the assembly code to place the value of 6 into the *AX* register since it has a semantics type of *fixed*. This value will then be saved onto the stack as a result of the generated *PUSH AX* statement.

In this case, the third statement, *tree_eval(one);*, will execute the *ADD CASE* tag, which generates the assembly code to place the value of *5+5* into *AX*. Code to pop the value of 6 off of the stack and into the *CX* register is generated as well, and the *IMULW* instruction multiplies *AX* by *CX* and places the result in the *AX* register.

A full listing of the parts of the *TREE_EVAL* procedure covered so far is shown in Listing 7. Notice that I call the *OPTIMIZE* procedure in the first line. As I showed earlier, *OPTIMIZE* will rearrange the tree structure, placing all simple values on the left-hand side of the tree structure.

Advanced translation techniques

To complete a grammar, we need to define a goal symbol that is a high-level definition of what the grammar is designed to accept. In this case the goal of the grammar is to accept Pascal statements, but a grammar rule like *Goal —> statement* specifies that the goal of my grammar is only to accept one single statement—not a very useful language.

To express the notion of an unlimited number of Pascal statements, I use a pair of recursive grammar rules:

```
statement-list —> statement-list
                    statement ;
                  —> statement ;
```

These rules specify that a *statement-list* may consist of either another *statement-list* followed by a single statement or just a single statement. The semicolon acts as the statement terminator. Figure 5 shows how these two productions can define an unlimited list of statements.

I can now use this definition for *statement-list* to define a Pascal *BEGIN-END* block:

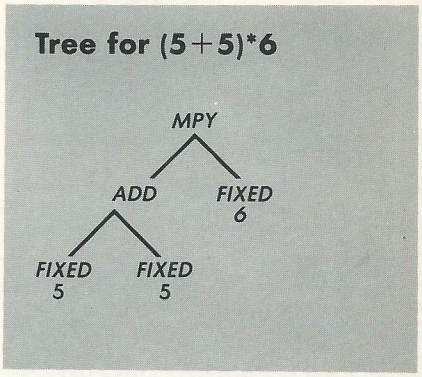
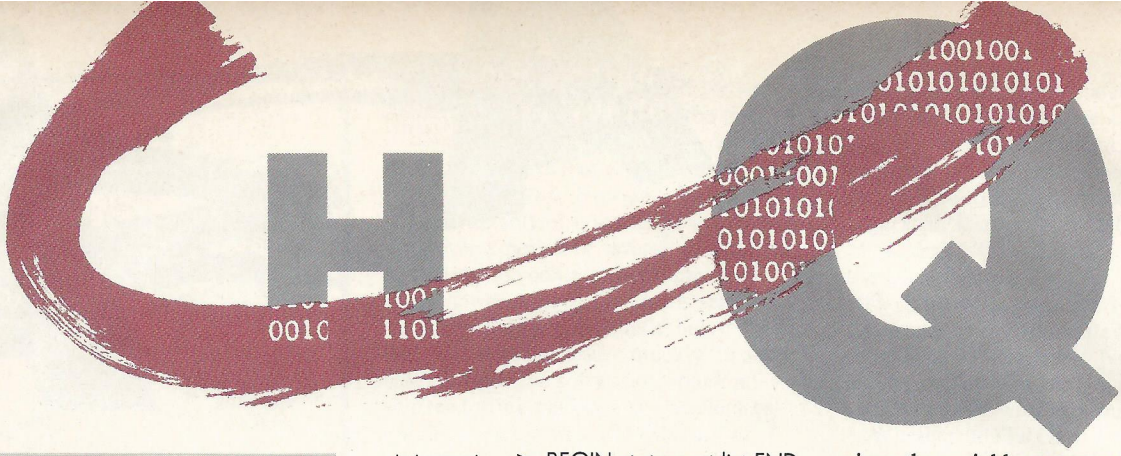


Figure 4.

```
Procedure TREE_EVAL(ROOT:semrecp);

begin
 optimize(root);
 with root^ do
   case semt of

  ASSIGN : begin
             if two^.semt = fixed then
               write('MOVW ',nameof(one),',',nameof(two))
             else begin
               tree_eval(two);
               write('MOVW ',nameof(one), ' ,AX');
             end;
           end;

    ADD: begin
           if is_simple(two) then begin
            tree_eval(one);
            write('ADD AX, ',nameof(two));
           end
           else begin
            tree_eval(two);
            write(PUSH AX);
            tree_eval(one);
            write(POP DX);
            write('ADD AX, DX');
           end
         end;

    IDENT, FIXED : write ('MOV AX, ', nameof(root));

    MPY: begin
           tree_eval(two);
           write('PUSH AX');
           tree_eval(one);
           write('POP CX');
           write('IMULW CX');
         end;
    end;
end;
```

Listing 7.



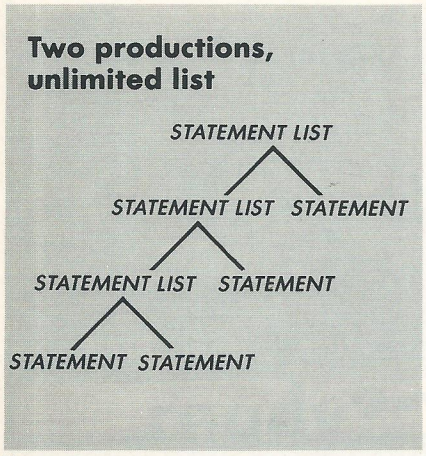


Figure 5.

```
goal -> var-declare statement ;
var-declare -> VAR varlist ; #VAR_DECL
varlist -> varlist , <identifier> #VAR_LIST
           -> <identifier>

statement -> <identifier> := expr #ASSIGN
          -> IF bool THEN statement ELSE statement #IF_THEN
          -> FOR <identifier> := <integer> TO <integer>  DO
statement #FOR_NEXT
          -> BEGIN statement-list END

statement-list -> statement-list statement ; #STATE_LIST
               -> statement ;

 bool -> primary boolop expr #BOOL_OP

 boolop -> >=
          -> <=
          -> =
          -> >
          -> <

  expr -> expr + term #ADD
          -> expr - term #SUB
          -> term

  term -> term * primary #MPY
          -> term / primary #DIV
          -> primary

primary -> ( expr )
          -> <integer>
          -> <string>
          -> <identifier>
```

Listing 8.

statement —> BEGIN statement-list END

The other grammar rules become more realistic with the addition of this grammar rule. For example, the statement:

FOR <identifier>:=<integer>
  TO <integer> DO statement

now expresses a true Pascal *FOR* loop:

```
FOR foobar:= 1 to 10 DO
begin
  foobar2:=foobar -1;
  foobar3:=foobar * 5;
end;
```

since the variable statement may be defined as a Pascal *BEGIN-END* block.

The full version of the Pascal-to-assembly language translator is designed to accept groups of entire Pascal functions, but for the sake of brevity I will limit the discussion here to the production of assembly code for a list of Pascal subset statements.

Given these limitations, the only missing piece from this Pascal subset grammar is a grammar definition for variable declaration. This definition is simple enough to implement with the rule:

var-declare —> VAR varlist ;

The variable *varlist* may be defined with the same recursive grammar rules used to define an unlimited list of statements. This time, however, I want to define an unlimited list of identifiers separated by commas:

varlist —> varlist , <identifier>
  —> <identifier>

Including a section for declaring variables may seem a little silly since the code I have written so far only supports *fixed* point variables, but keep in mind that the full version of the program offers support for floating point and string variables as well as the Pascal *RECORD* structure.

Since I am working with a very limited subset of Pascal that does not support functions, the goal of this grammar is to define a variable declaration section followed by an unlimited list of Pascal statements. Using some of the rules I have already defined, I may define the goal symbol for this grammar as:

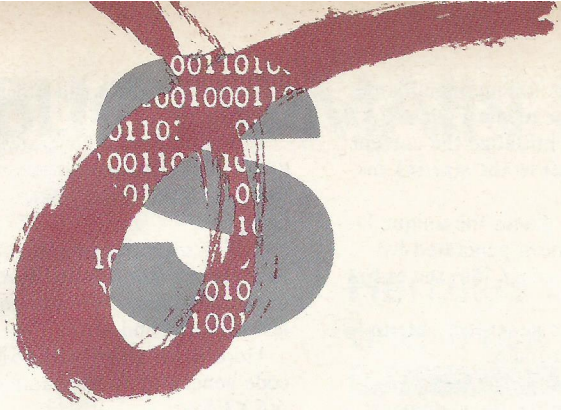
goal —> var-declare statement ;

The full grammar for this Pascal subset is shown in Listing 8.

The structure of the programs this grammar is written to accept is a variable declaration block:

VAR foobar,foobar2,foobar3;

followed by a single statement or group of statements enclosed within a *BEGIN-END* block. Let's look at the process of



tree evaluation for a multistatement *FOR-NEXT* tree.

Evaluating the FOR-NEXT tree

Once you have mastered some of the basic tree-walking concepts, it is simple to write code to evaluate even the most complex trees.

For example, the tree constructed by the generated parsing program for the input:

```
FOR I:= 5 to 10 DO
begin
   foobar:=foobar + 1;
   foobar2:= foobar + 3;
end;
```

is shown in Figure 6. Note that the semantics types appear in italics while the actual values are in a standard font.

The evaluation scheme for this tree is actually much simpler than it appears since I have already written code to handle all but two of the semantics types listed on the tree. I will need to evaluate both sides of the *STATE_LIST* tree, but I do not know the contents of either leaf. Thus the *CASE* tag:

```
STATE_LIST: begin
          eval(one);
          eval(two);
   end;
```

will allow other *CASE* tags to do the real work of evaluation for each of the two leaves on the *state_list* tree. Look at the structure of the *statement-list* tree in Figure 6 to see how this *CASE* tag will work for trees with more than two statements.

The *CASE* tag to produce assembly code for the *FOR_NEXT* tag is shown in Listing 9. The procedure *GEN_LABEL* called in line 1 generates a unique label name. Leaf 1 of a *FOR_NEXT* tree is always an identifier,

Complex tree

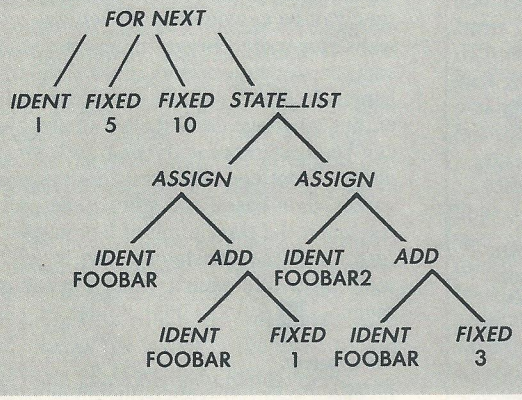


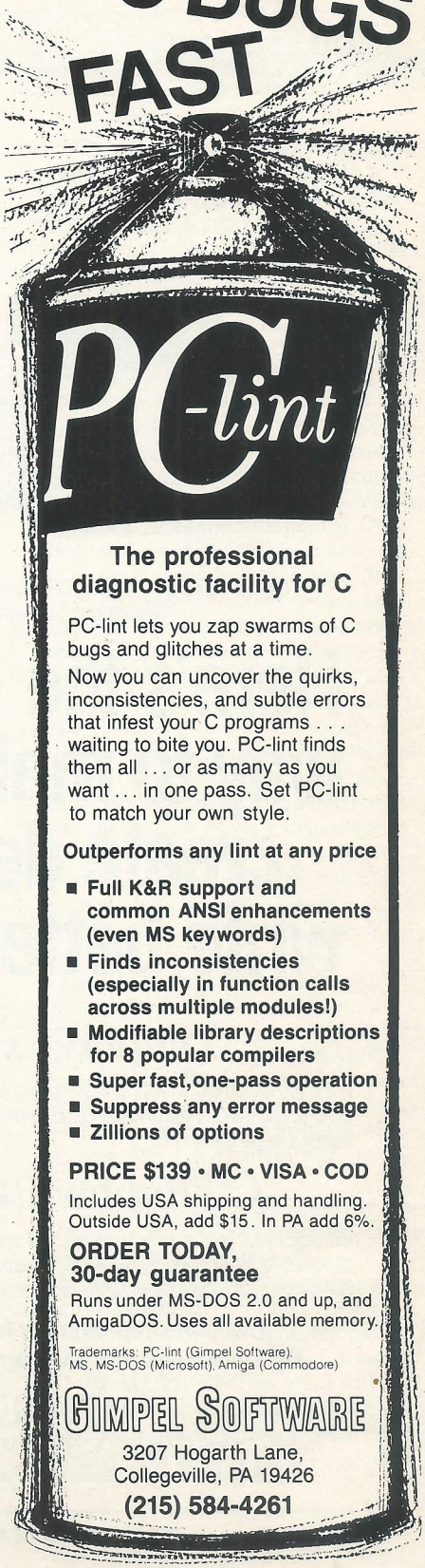
Figure 6.

```
FOR_NEXT: begin
            label_one:=gen_label;
            write('MOVW ', nameof(one),', ',nameof(two));
            write(label_one, ' EQU $');
            write('INCW ', nameof(one));
            tree_eval(four);
            write('CMPW ', nameof(one), ', ',nameof(three));
            write('JLE ' , label_one);
           end;
```

Listing 9.

#1 Lint for MS-DOS

KILLS C BUGS FAST



The professional diagnostic facility for C

PC-lint lets you zap swarms of C bugs and glitches at a time.

Now you can uncover the quirks, inconsistencies, and subtle errors that infest your C programs . . . waiting to bite you. PC-lint finds them all . . . or as many as you want . . . in one pass. Set PC-lint to match your own style.

Outperforms any lint at any price

- Full K&R support and common ANSI enhancements (even MS keywords)
- Finds inconsistencies (especially in function calls across multiple modules!)
- Modifiable library descriptions for 8 popular compilers
- Super fast, one-pass operation
- Suppress any error message
- Zillions of options

PRICE $139 • MC • VISA • COD

Includes USA shipping and handling. Outside USA, add $15. In PA add 6%.

ORDER TODAY, 30-day guarantee

Runs under MS-DOS 2.0 and up, and AmigaDOS. Uses all available memory.

Trademarks: PC-lint (Gimpel Software). MS, MS-DOS (Microsoft). Amiga (Commodore)

GIMPEL SOFTWARE

3207 Hogarth Lane,
Collegeville, PA 19426

(215) 584-4261

leaf 2 is always an integer representing the starting index, leaf 3 is an integer representing the ending index, and leaf 4 is the statement list that constitutes the body of the loop:

```
FOR I:= 5 TO 10          STLIST
    ^     ^     ^          ^
   one   two   three      four
```

Here is a line-by-line analysis of this code:

1. The procedure *GEN_LABEL* called in line 1 generates a unique label name.

2. The second line of the *FOR-NEXT* tag creates code to initialize the current index (*I* in this case) to the starting index (5 in this case).

3. The third line places the unique label name that has been generated by procedure *GEN_LABEL* into the assembly code.

4. The fourth line generates code to increment the current index.

5. The fifth line calls the tree evaluator to generate code for the statements that constitute the body of the loop. Several recursive calls to evaluate the *ADD* and *ASSIGN* trees will be executed in this case.

6. The sixth line generates the 8086/8088 code to compare the current index with the ending index (10 in this case).

7. The seventh line generates the code to execute a jump to the label generated in the third line if the current index is less than the end index.

Here is the actual CHASM assembly code generated by the sample *FOR-NEXT* loop:

```
        MOVW I, 5
LABEL1  EQU $
        INCW I
        MOVW AX, FOOBAR
        ADDW AX, 1
        MOVW FOOBAR, AX
        MOVW AX, FOOBAR2
        ADDW AX, 3
        CMPW I, 10
        JLE LABEL1
```

Other applications

This article has provided an interesting but very limited sample of language-intensive programming techniques. Without a translator writer, programming a language-to-language converter of the type I have illustrated would have required about two weeks of additional effort. But language translators are by no means the only applications that can be developed with a translator writer.

For example, the translator writer automatically generates a syntax checker from any grammar file, and programming a syntax checker for a complex language like Pascal or C is no small task.

Language development is another very important application whose development time can be cut significantly with a translator writer. Programmers designing compilers or interpreters could obviously benefit from the use of a translator writer, but language development of any type can be accelerated with the translator writer. Front-ends for operating systems, games, data bases, and spreadsheets all rely upon the translation of language forms, and the translator writer is the tool that will let your computer speak in tongues.

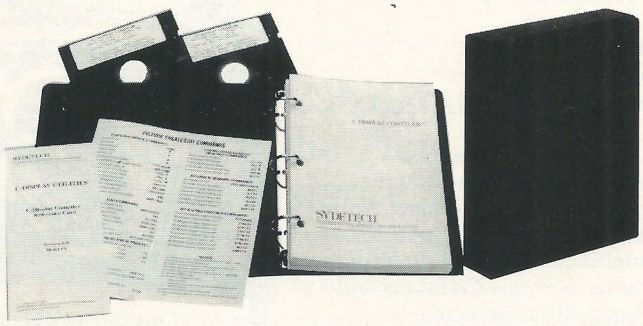
References

Barrett, W., R. Bates, D. Gustafson, J. Couch. *Compiler Construction Theory and Practice,* SRA Publishing, 1986.

QCAD SYSTEMS, Tiny Pascal and QPARSER+, 3333A Octavius Dr., Santa Clara, Calif. 95054, (408) 727-6671

Whitman, David. CHASM. 136 Wellington Terrace, Lansdale, Pa. 19446, (215) 641-7114.

Wirth, Niklaus. A Portable Pascal Compiler.



DisplayExpress™ and C-Display Librarian™ come with superb, professionally written documentation

Here's the quick, easy way to make screens, create demos and write better programs

DisplayExpress™ $99

Helps you design the screens as easily as you would type a letter. It's fast, user-friendly and easy to learn. And it allows you to create demos, presentations and continuously running shows without writing a single line of code.

C-Display Librarian™ $99

Increases your productivity and helps you write better applications. 96 functions specifically designed for BIOS video, screen editing, data entry and validation and bit mapped graphics. Compatible with Lattice C and Microsoft C, Fortran and Pascal compilers.

Available immediately for IBM and TI computers.
By mail or write for the name of the dealer nearest you.

SYDETECH
SYSTEM DEVELOPMENT TECHNOLOGIES
43-23 Colden St., Suite 17C, Flushing, NY 11355, (718) 886-0221

CIRCLE 34 ON READER SERVICE CARD

Artwork: Steve Campbell

...thanks!

When we founded Programmer's Connection in 1984, we dedicated ourselves to providing high quality personal service to every customer. Since then, we've quickly grown to be the leading independent dealer in this industry.

CELEBRATE WITH US

To celebrate our success, we're offering special sale prices on many of our products. We're also featuring some select, high-quality products on this page that represent exceptional values.

As we now enter our fourth year of business, we'd like to say "thanks!" to all of you who gave us the opportunity to serve you. And to those of you who haven't yet discovered our low prices and quality service, we extend a hearty invitation to give us a try You'll be glad you did!

Turn the page for our latest advertised price list.

Sale prices effective through June 30, 1987.

Call or write for our FREE comprehensive price guide.

TURBOsmith by Visual Age is a powerful debugging environment for Turbo Pascal featuring **automatic**, multi-window readout of your source programs and variables. Trace and breakpoint with single-keystroke commands and watch the values of variables and expressions change. Supports Overlays, .COM and chain files. Synchronized Machine Code window has built-in assembler for in-line work.

List *$69* **Ours $59**

PERSONAL COBOL by Micro Focus is a fully-integrated program development system for creating sophisticated applications on your IBM Personal Computer or Personal System/2. This versatile package comes complete with a full-screen text editor, source code generator for screens, syntax checker, automatic file handling and an interactive source-level debugging facility.

List *$149* **Sale $99**

***PURCHASE A SUPPORTED C COMPILER** and receive 50% off any of the following libraries (when purchased on the same order):

	List	Without Language	*With Language
Data Management Consultants			
Zview Screen Mgmt Library	$245	$139	**$122.50**
Essential Software			
C Utility Library	185	119	**92.50**
Essential Graphics	250	183	**125.00**
Essential Comm Library	185	125	**92.50**
with Breakout Debugger	250	189	**125.00**

QUALITY. *We carry the finest selection of the best programmer's development tools specifically for IBM Personal Computers and compatibles. They are the latest versions and most come with return guarantees or evaluation periods.*

SUPPORT. *Our courteous, knowledgeable, non-commissioned salespeople are always ready to assist you. Our experienced technical consultants can answer questions about products and provide sound, unbiased advice.*

PRICE. *UPS Ground shipping is FREE to all U.S. customers. There are no extra charges for credit cards, CODs, purchase orders, sales tax (except Ohio) or special handling (except for export preparation). Quite simply, the prices on the next two pages are all you pay.*

INTEGRITY. *We're very proud of the trust we've earned from our customers and pledge always to be worthy of it.*

PROGRAMMER'S QUALITY INTEGRITY Service SUPPORT PRICE CONNECTION

arity products

Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	50	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	50	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77

ai - expert systems

1st-CLASS *by Programs in Motion*	495	399
AutoIntelligence *by IntelligenceWare*	990	739
ExpertEDGE Advanced *by Human Edge*	2500	CALL
ExpertEDGE Professional *by Human Edge*	5000	CALL
Experteach II *by IntelligenceWare*	475	339
EXSYS *Development Software by EXSYS*	395	309
EXSYS Runtime System	600	469
Insight 1 *by Level Five Research*	95	75
Insight 2+ *by Level Five Research*	485	379
Intelligence/Compiler *IntelligenceWare*	990	739
Logic-Line Series 1 *by Thunderstone*	90	85
Logic-Line Series 2 *by Thunderstone*	125	115
Logic-Line Series 3 *by Thunderstone*	150	139

ai - lisp language

GCLISP *Golden Common LISP by Gold Hill*	495	CALL
GCLISP 286 Developer *by Gold Hill*	1190	CALL
IQCLISP *by Integral Quality*	300	CALL
IQLISP *by Integral Quality*	270	CALL
Microsoft LISP *Common LISP*	250	149
QNIAL *Combines LISP & APL by NIAL Systems*	375	339
Starter QNIAL *by NIAL Systems* *New*	99	89
TransLISP *from Solution Systems*	95	CALL
TransLISP PLUS *from Solution Systems*	195	CALL

ai - prolog language

APT *Active Prolog Tutor from Solution Systems*	65	CALL
LPA microPROLOG *All Varieties*	CALL	CALL
MPROLOG Language Primer *LOGICWARE*	50	45
MPROLOG P500 *by LOGICWARE*	495	395
MPROLOG P550 *by LOGICWARE*	220	175
Prolog-86 *from Solution Systems*	125	CALL
Prolog-86 Plus *from Solution Systems*	250	CALL
Turbo PROLOG *by Borland Intl*	100	63
Turbo PROLOG Toolbox *by Borland Intl*	100	64

ai - smalltalk language

Smalltalk/V *by Digitalk*	99	84
EGA/VGA Color Option	100	85
Goodies Diskette	49	45
Smalltalk/Comm	49	42

ai - texas instruments

PC Scheme *Lisp*	95	84
Personal Consultant Easy	495	435
Personal Consultant Plus	2950	2589
Personal Consultant Runtime	95	84

apl language

APL*PLUS/PC *by STSC*	595	424
APL*PLUS/PC Spreadsheet Mgr *by STSC*	195	139
APL*PLUS/PC Tools Vol 1 *by STSC*	295	199
APL*PLUS/PC Tools Vol 2 *by STSC*	85	58
ATLAS*GRAPHICS *by STSC* *New*	CALL	CALL
Financial/Statistical Library *by STSC*	275	189
Pocket APL *by STSC*	95	69
STATGRAPHICS *by STSC*	795	579

assembly language

386 ASM/LINK *Cross Asm by Phar Lap*	495	389
8088 Assembler *w/ Z-80 Translator by 2500 AD*	100	89
Advanced Norton Utilities *by Peter Norton* *New*	150	99
ASMLIB *Function Library by BC Assoc*	149	125
asmTREE *B-Tree Dev System by BC Assoc*	395	339
Cross Assemblers *Various by 2500 AD*	CALL	CALL
Microsoft Macro Assembler	150	93
Norton Utilities *by Peter Norton*	100	59
screenplay *by Flexus*	100	79
Turbo EDITASM *by Speedware*	99	84
Uniware Cross Assemblers *Various by SDS*	CALL	CALL
Visible Computer: 8088 *Software Masters*	80	64

basic language

87 Software Pak *by Hauppauge*	180	149
EXIM Services Toolkit *by EXIM*	50	45
Finally *by Komputerwerks*	99	85
Inside Track *from Micro Help*	65	51
MACH 2 *by Micro Help*	69	55
Microsoft QuickBASIC Compiler	99	63
Peeks 'n Pokes *from MicroHelp*	45	37
Professional BASIC *by Morgan*	99	68
8087 Math Support	50	42
QuickPak *by Crescent Software*	69	59
Scientific Subroutine Library *by Peerless*	125	99
Stay-Res *by MicroHelp*	95	73
True Basic *w/ Run-time*	245	179
True Basic	150	97
Run-time Module	150	97
Various Utilities	50	41
Turbo BASIC Compiler *by Borland Intl*	100	64

blaise products

ASYNCH MANAGER *Specify C or Pascal*	175	119
C TOOLS PLUS	175	119
EXEC *Program Chainer*	95	65
LIGHT TOOLS *for Datalight C*	100	CALL
PASCAL TOOLS	125	94
PASCAL TOOLS 2	100	CALL
PASCAL TOOLS & TOOLS 2	175	119
RUNOFF *Text Formatter*	50	39
TURBO ASYNCH PLUS	100	CALL
TURBO POWER TOOLS PLUS	100	CALL
VIEW MANAGER *Specify C or Pascal*	275	179

borland products

EUREKA *Equation Solver*	100	64
Reflex & Reflex Workshop	200	128
Reflex *Data Base System*	150	89
Reflex Workshop	70	45
Sidekick & Traveling Sidekick	125	85
Sidekick	85	57
Traveling Sidekick	70	45
Superkey *New Version*	100	64
Turbo BASIC Compiler	100	64
Turbo C Compiler *New*	100	64
Turbo Database Toolbox	70	41
Turbo Editor Toolbox	70	41
Turbo Gameworks Toolbox	70	41
Turbo Graphix Toolbox	70	41
Turbo Jumbo Pack *Combination Package* *New*	300	219
Turbo Lightning	100	64
Turbo PASCAL Numerical Methods Toolbox	100	64
Turbo PASCAL and Tutor	125	85
Turbo PASCAL	100	64
Turbo Tutor	40	24
Turbo PROLOG Compiler	100	63
Turbo PROLOG Toolbox	100	64
Word Wizard	70	47
Word Wizard and Turbo Lightning	150	94

c++

C++ *by Guidelines w/ version 1.1 kernel*	195	172
PforCe++ *Function Library by Phoenix*	395	225

c compilers

C86PLUS *by Computer Innovations*	497	CALL
Datalight C Compiler *Small Model*	60	43
Datalight Developer Kit	99	74
Datalight Optimum-C	139	99
DeSmet C *w/ Debugger & Large Case*	209	184
DeSmet C *w/ Debugger Only*	159	138
Eco-C *with Graphics by Ecosoft*	125	83
Lattice C Compiler *vers. 3.2 from Lattice* *Sale*	500	249
Mark Williams Let's C Combo Pack *New Version*	125	99
Let's C *Compiler* *New Version*	75	54
csd *Source Level Debugger* *New Version*	75	54
Microsoft C *with CodeView*	450	269
Turbo C Compiler *by Borland Intl* *New*	100	64
Uniware 68000/10/20 Cross Compiler *by SDS*	CALL	CALL

c interpreters

C-terp *by Gimpel, Specify compiler*	300	219
C Trainer *with Book by Catalytix*	122	87
Instant C *by Rational Systems* *Sale*	500	349
Introducing C *by Computer Innovations*	125	CALL
Run/C *by Age of Reason*	120	79
Run/C Professional *by Age of Reason*	250	157

c utilities

C to dBase *by Computer Innovations*	150	CALL
c-tree & r-tree Combo *by FairCom* *Sale*	650	449
c-tree *ISAM File Manager* *Sale*	395	269
r-tree *Report Generator* *Sale*	295	199
C Windows *by Syscom*	100	85
C Wings *by Syscom*	50	43
CI ROMPac *by Computer Innovations*	195	CALL
dBx *dBase to C Translator by Desktop AI*	350	299
with Library Source Code	550	469
Various Support Utilities	CALL	CALL
EASY SCREEN *by Retail Mgmt Systems* *New*	225	199
Entelekon Products	CALL	CALL
Flash-up Windows *by Software Bottling*	90	78
GraphiC *Color version by Sci Endeavors*	350	282
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
HALO Development Pkg *for Microsoft* *Sale*	595	359
The HAMMER *by OES Systems*	195	129
PANEL *Forms Management by Roundhill*	295	CALL
PANEL Plus *by Roundhill* *New*	495	CALL
PC Lint *by Gimpel Software*	139	99
PLOTHI *by Sutrasoft*	175	159
PLOTHP *by Sutrasoft*	175	159
Professional C Windows *by Washburn*	89	79
Scientific Subroutine Library *by Peerless*	175	128
screenplay *for C by Flexus*	CALL	CALL
Vitamin C *by Creative Programming*	225	158
VC Screen *Forms Designer*	100	79
Zview *by Data Mgmt Consultants* *See First Page*	245	139

cobol language

COBOLspII *by Flexus* *New*	CALL	CALL
FPLIB *for Realia COBOL by BC Associates* *New*	149	129
Micro Focus COBOL *See Micro Focus Section*		
Microsoft COBOL *See Microsoft Section*		
Realia COBOL with RealMENU *New*	1145	899
Realia COBOL	995	783
RealCICS	995	783
RM/COBOL *by Ryan-McFarland*	950	CALL
RM/COBOL 85 *by Ryan-McFarland*	1250	CALL
screenplay *for COBOL by Flexus*	175	129

debuggers & profilers

386 DEBUG *Cross Debugger by Phar Lap*	195	129
Advanced Trace-86 *by Morgan Computing*	175	115
CI Probe *by Computer Innovations*	225	CALL
Codesifter *Profiler by David Smith*	119	85
Codesmith-86 *by Visual Age*	145	98
DSD87 *by Soft Advances*	125	99
MiniProbe *by Atron*	395	CALL
Periscope I *with Board by Periscope*	345	289
Periscope II *with NMI Breakout Switch*	175	139
Periscope II-X *Software only*	145	105
Periscope III *w/ Advanced Board* *New*	995	CALL
The PROFILER *with Source Code by DWB*	125	89
TURBOsmith *for Turbo Pascal* *See First Page*	69	59
The WATCHER *Profiler by Stony Brook*	60	51

dos utilities

Command Plus *by ESP Software*	80	69
FANSI-CONSOLE *by Hersey Micro*	75	62
Norton Commander *by Peter Norton*	75	55
Scroll & Recall *by Opt-Tech Data*	69	59
Taskview *by Sunny Hill Software*	80	55

essential products

See First Page for Special Offers.

C Essentials *by Essential Software*	100	75
C Utility Library	185	119
Essential Comm Library *with Debugger*	250	189
Essential Comm Library *Software Only*	185	125
Breakout Debugger Only *Any language*	125	89
Essential Graphics	250	183

forth language

CFORTH *Native Code Compiler by LMI*	300	229
Forth/83 Metacompiler *Specify Target*	750	599
PC/Forth *by Laboratory Microsystems*	150	109
PC/Forth+ *by Laboratory Microsystems*	250	199
Advanced Color Graphics Support	100	74
Enhanced Graphics Support	200	148
Intel 8087 Support	100	74
Interactive Symbolic Debugger	100	74
Native Code Optimizer	200	148
Software Floating Point	100	74
UR/Forth *by LMI*	350	279
Object Module Libraries	500	395

fortran language

50 MORE: FORTRAN *by Peerless Engr*	125	95
ACS Time Series *Alpha Computer Service*	495	389
Essential Graphics *by Essential Software*	250	183
For-Winds *Alpha Computer Service*	90	69
Forlib-Plus *Alpha Computer Service*	70	44
FORTLIB *by Sutrasoft*	125	109
FORTRAN Addendum *by Impulse Engr*	95	85
FORTRAN Addenda *by Impulse Engr*	165	138
GRAFLIB *by Sutrasoft*	175	159
HALO *by Media Cybernetics*	300	205
I/O PRO *by MEF Environmental*	149	129
Microcompatibles Combo Package	240	215
Grafmatic	135	117
Plotmatic	135	117
Microsoft FORTRAN *w/ CodeView*	450	269
No Limit *by MEF Environmental*	129	109
Numerical Analyst *by MAGUS*	295	CALL
PANEL *by Roundhill Computer Systems*	295	CALL
PLOTHI *by Sutrasoft*	175	159
PLOTHP *by Sutrasoft*	175	159
RM/FORTRAN *Ryan-McFarland*	595	CALL
RTC PLUS *Fortran to C by Cobalt Blue*	325	CALL
Scientific Subroutine Lib *by Peerless*	175	128
Statistician *Alpha Computer Service*	295	235
Statlib.GL: *by PSISystems*	295	239
Statlib.TSF: *by PSISystems*	295	239
Strings & Things *Alpha Computer Service*	70	45

greenleaf products

Greenleaf Comm Library	185	125
Greenleaf Data Windows	225	157
with Source Code	450	289
Greenleaf Functions	185	125

help utilities

HELP/Control *by MDS*	125	99
On-line Help *from Opt-Tech*	149	99
SoftScreen/HELP *by Dialectic Systems*	195	149

lattice products

Lattice C Compiler *ver 3.2 from Lattice* *Sale*	500	249
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord *Function Library*	150	95
with Source Code	300	179
C-Sprite *Source Level Debugger*	175	119
Curses *Screen Manager*	125	85
with Source Code	250	169
dBC *Specify dBC II or dBC III*	250	169
with Source Code	500	356
dBC III Plus	750	CALL
with Source Code	1500	CALL
LMK *Make Facility*	195	138
RPG II Combo *All three items below*	1100	875
RPG II Compiler *No Royalties*	750	625
RPG II SEU *Screen Entry Utility*	250	199
RPG II Sort/Merge	250	199
RPG II Screen Design Aid Utility	350	309
SecretDisk *File Encryption Utility*	120	88
SideTalk *Resident Communications*	120	88
SSP/PC *Scientific Subroutine Library* *Sale*	350	229
Text Management Utilities	120	88
TopView Toolbasket *Function Library*	250	178
with Source Code	500	356

metagraphics products

LightWINDOWS/C *for Datalight C* *New*	95	79
MetaWINDOWS *No Royalties*	185	109
MetaFONTS	80	58
MetaWINDOWS/Plus	285	184
MetaFONTS/Plus	235	184
TurboWINDOW *Graphics/ Windows for Turbo Pascal*	80	58

micro focus products

Micro Focus Level II COBOL *w/Animator* *New*	495	395
Level II COBOL	349	279
Level II Animator	195	155
Micro Focus Level II COBOL/ET for UNIX *New*	CALL	CALL
Micro Focus Personal COBOL *See First Page*	149	99
Micro Focus Professional COBOL	2000	1595
Micro Focus VS COBOL/XENIX *New*	1495	1195

Support Products:			
COBOL/IQ *Ad hoc Report Writer*	*New*	495	395
COBOL/IQ *for DOS 3.X Networks*	*New*	995	795
FORMS-2		295	235
SOURCEWRITER		995	795

microport products

System V/AT *by Microport Systems*	*Sale*	549	459
Runtime System *(Operating System)*		199	189
Software Development System		249	235
Text Preparation System		199	189
User Upgrade *3 to Unlimited Users*		249	235

microsoft products

Microsoft BASIC Interpreter *for XENIX*		350	209
Microsoft C *with CodeView*		450	269
Microsoft COBOL *w/ COBOL Tools*	*New Version*	700	429
for XENIX		995	609
Microsoft FORTRAN *w/ CodeView*		450	269
for XENIX		695	419
Microsoft Learning DOS		50	36
Microsoft LISP *Common LISP*		250	149
Microsoft MACH 10 *w/ Mouse & Windows*		549	369
Microsoft MACH 10 *Board only*		399	279
Microsoft Macro Assembler		150	93
Microsoft Mouse *Bus Version*		175	114
Microsoft Mouse *Serial Version*		195	124
Microsoft muMath *Includes muSIMP*		300	179
Microsoft Pascal Compiler		300	179
for XENIX		695	419
Microsoft QuickBASIC Compiler		99	63
Microsoft Sort		195	125
Microsoft Windows		99	63
Microsoft Windows Development Kit		500	299

modula-2 language

IOTools *by Rhodes Associates*		80	69
with Source Code		950	CALL
MODULA-2 Apprentice Pkg by LOGITECH		99	79
MODULA-2 Magic Pkg by LOGITECH		99	79
MODULA-2 ROM Pkg & Cross RT Debugger		299	239
MODULA-2 Window Pkg by LOGITECH		49	39
MODULA-2 Wizard's Pkg by LOGITECH		199	159
REPERTOIRE *for MODULA-2 by PMI*		89	79
Object Code Only		19	15

mouse products

LOGIMOUSE BUS *with PLUS Pkg by LOGITECH*		119	98
with PLUS & PC Paintbrush		149	119
with PLUS & CAD Software		189	153
with PLUS & CAD & Paint		219	179
LOGIMOUSE C7 *with PLUS Pkg, Specify Connector*		119	98
with PLUS & PC Paintbrush		149	119
with PLUS & CAD Software		189	153
with PLUS & CAD & Paint		219	179

other languages

CCS MUMPS *Single-User by MGlobal*		60	50
CCS MUMPS *Single-UserMulti-Tasking*		150	129
CCS MUMPS *Multi-User*		450	359
Janus/ADA C Pak *by R&R Software*		95	84
Janus/ADA D Pak *by R&R Software*		900	769
Janus/ADA ED Pak *by R&R Software*		395	CALL
Marshal Pascal *by Marshal Language Systems*		189	155
Personal REXX *by Mansfield Software*		125	99
SNOBOL4+ *by Catspaw*		95	80

other products

Dan Bricklin's Demo Pgm *Software Garden*		75	57
Dan Bricklin's Demo Tutorial	*New*	50	45
Disk Optimizer *by Softlogic Systems*		60	55
FASTBACK *by 5th Generation Systems*		179	133
Informix *All Varieties by Informix*		CALL	CALL
Instant Replay *by Nostradamus*	*New Version*	CALL	CALL
MKS Toolkit *with vi Editor by MKS*		139	99
Net-Tools *by BC Associates*		149	129
OPT-Tech Sort *by Opt-Tech Data Proc*		149	99
PC/TOOLS *by Custom Software*	*New*	49	45
Screen Machine *by MicroHelp*	*New*	79	59
VTEK *Term Emulator by Sci Endeavors*		150	129

phoenix products

Pasm86 *Macro Assembler Version 2.0*		195	108
Pdisk *Hard Disk & Backup Utility*		145	99
Pfantasy Pac *Phoenix Combo*		995	619
Pfinish *Execution Profiler*		395	225
Pfix86plus *Symbolic Debugger*		395	225
PforCe *Comprehensive C Library*		395	225
PforCe++ *Library for Guidelines C++*		395	225
Plink86plus *Overlay Linker*		495	299
Pmaker *Make Utility*		125	78
Pmate *Macro Text Editor*		195	108
Pre-C *Lint Utility*		295	154
Ptel *Binary File Transfer Program*		195	108

polytron products

PolyBoost *The Software Accelerator*		80	64
PolyDesk III	*New*	99	72
PolyDesk III Archivist	*New*	50	42
PolyDesk III Cryptographer	*New*	50	42
PolyDesk III Talk	*New*	70	52
PolyLibrarian *Library Manager*		99	73
PolyLibrarian II *Library Manager*		149	109
PolyMake *UNIX-like Make Facility*		149	109
PolyShell		149	109
Polytron C Beautifier		50	42
Polytron C Library I		99	72
Polytron PowerCom *Communications*		139	105
PolyXREF *Complete Cross Ref Utility*		219	169
PolyXREF *One language only*		129	99
PVCS Corporate *Version Control System*		395	309
PVCS Personal		149	109
PVMFM *Polytron Virtual Memory File Mgr*	*New*	CALL	CALL

program mgmt utilities

Compact Source Print *by Aldebaran*		55	44
Interactive EASYFLOW *by Haventree*		150	125
PrintQ *by Software Directions*		89	84
Quilt Computing Combo Package	*New Version*	250	199
QMake *Program Rebuild Utility*		99	79
SRMS *Software Revision Mgmt Sys*	*New Version*	185	159
Source Print *by Aldebaran Labs*		75	59
TLIB *by Burton Systems Software*		100	89
Tree Diagrammer *by Aldebaran Labs*		55	48

raima products

dbQUERY *Single-User Query Utility*		195	129
Single-User with Source Code	*Sale*	495	329
Multi-User	*Sale*	495	329
Multi-User with Source Code	*Sale*	990	659
dbVISTA *Single-User DBMS*		195	129
Single-User with Source Code	*Sale*	495	329
Multi-User	*Sale*	495	329
Multi-User with Source Code	*Sale*	990	659

sco products

Complete XENIX System V *by SCO*		1295	994
Development System		595	499
Operating System *Specify XT or AT*		595	499
Text Processing Package		195	144
Networks for XENIX *by SCO*		595	495
SCO Professional *Lotus clone for XENIX*		795	595

softcraft products

Btrieve *ISAM Mgr with No Royalties*		245	184
Xtrieve *Query Utility*		245	184
Report Option *for Xtrieve*		145	99
Btrieve/N *for Networks*		595	454
Xtrieve/N		595	454
Report Option/N *for XtrieveN*		345	269

text editors

Brief & dBrief Combo *from Solution Systems*		250	CALL
Brief *Programmer's Text Editor*		195	CALL
dBrief *Customizes Brief for dBase III*		95	CALL
Epsilon *Emacs-like editor by Lugaru*		195	147
KEDIT *by Mansfield Software*		125	98
Micro/SPF *by Phaser Systems*		175	139
PC/VI *by Custom Software Systems*		149	99
SPF/PC *by Command Technology Corp*	*New Version*	245	CALL
Vedit *by CompuView*		150	98
Vedit Plus *by CompuView*		185	128

turbo pascal utilities

ALICE *Interpreter by Software Channels*		95	66
DOS/BIOS & Mouse Tools *by Quinn-Curtis*		75	67
Flash-up Windows *by Software Bottling*		90	78
MACH 2 *for Turbo Pascal by Micro Help*	*New*	69	55
MetraByte D/A Tools *by Quinn-Curtis*		100	89
Science & Engrg Tools *by Quinn-Curtis*		75	67
Screen Sculptor *by Software Bottling*		125	91
screenplay *for Turbo Pascal by Flexus*		100	79
Speed Screen *by Software Bottling*		35	32
System Builder *by Royal American*		100	CALL
IMPEX *Query Utility*		75	CALL
Report Builder		75	CALL
TDebugPLUS *by TurboPower Software*		60	49
Turbo EXTENDER *by TurboPower Software*		85	64
Turbo Professional *by Sunny Hill*		70	45
TurboHALO *from IMSI*		129	98
TurboPower Utilities *by TurboPower*		95	78
TurboRef *by Gracon Services*		50	45
TURBOsmith *Visual Age Debugger*	*See First Page*	69	59
Universal Graphics Library *by Quinn-Curtis*	*New*	150	119

wendin products

Operating System Toolbox	*Rebate Offer*	99	75
PCNX *Operating system*	*Rebate Offer*	99	75
PCVMS *Similar to VAXVMS*	*Rebate Offer*	99	75
XTC *Text Editor w/ Pascal source*	*Rebate Offer*	99	75

xenix/unix products

Btrieve *ISAM File Mgr by SoftCraft*		595	454
C-terp *by Gimpel, Specify compiler*		498	379
c-tree *ISAM Mgr by FairCom*	*Sale*	395	269
dbVISTA *See Raima Section*			
dBx *with Library Source by Desktop AI*		550	489
DOSIX *Console Version by Data Basics*		399	CALL
DOSIX *User Version by Data Basics*		199	CALL
Informix *All Varieties by Informix*		CALL	CALL
Lyrix *by Informix*		595	449
Micro Focus Level II Compact COBOL		1000	CALL
Forms-2		400	CALL
Level II ANIMATOR		600	CALL
Microport Products *See Microport Section*			
Microsoft Products *See Microsoft Section*			
PANEL Plus *by Roundhill Computer Systems*	*New*	CALL	CALL
REAL-TOOLS *Binary Version by PCT*		149	89
Library Source Version		399	289
Complete Source Version		499	369
RM/COBOL *by Ryan-McFarland*		1250	CALL
RM/FORTRAN *by Ryan-McFarland*		750	CALL
SCO Products *See SCO Section*			

LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and telephone number.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. POs with net 30-day terms are available to qualified US accounts only.

FOREIGN ORDERS

Shipping charges for foreign and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. Foreign orders (except Canada), please include an additional $10 for customs form preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection
136 Sunnyside Street
Hartville, OH 44632

CALL TOLL FREE

U S 800-336-1166
CANADA 800-225-1166
OHIO & ALASKA
(Call Collect) 216-877-3781
TELEX 9102406879

FOREIGN 216-877-3781
CUSTOMER SERVICE 216-877-1110

Hours: Weekdays 8:30 AM to 8:00 PM EST.

Ohio customers add 6% state sales tax.
Prices are subject to change without notice.
Copyright Programmer's Connection, Inc., 1987.

programmer's connection

CIRCLE 35 ON READER SERVICE CARD

C BRICKLIN RUN

Data Entry • Menus • Windows • Prototyping • Database • Toolkit

C-scape

■ Total Screen Control/Easy to Use

C-scape is a combination screen generator and library of screen I/O functions. Written for C programmers, C-scape brings a proven approach to the need for an easy-to-learn and use, but truly powerful and flexible screen management tool.

C-scape's kernel is your most powerful ally. Without requiring parameters you'll never use, it allows you to create tailored functions with ease and simplicity. Each key is individually definable. If you know *printf*(), you can use C-scape. C-scape's kernel provides a veritable screen design and construction toolkit to rewrite our functions or to write your own.

■ Most Powerful Prototyping Available

C-scape offers a unique approach to prototyping your software. You may use **Dan Bricklin's Demo Program** to create, edit, and view your screens (you can even capture existing screens from other programs), and then use C-scapes's **demo2c** utility to convert each screen to code.

You can design each screen with attributes such as colors, menu selections, data entry fields (including type, validation, and field naming), masking, and text, and then automatically convert the entire screen to code.

■ Powerful Function Library

Use C-scape's functions for Lotus-like, pull-down, or your own menu designs, automatic scrolling, pop-up windows (number limited only by RAM), logical colors, help, time and date, yes/no, tickertape fields, secure and protected fields, and many others, to turn your demo into a fully functioning and complete program in a fraction of the time spent coding screens from scratch.

C-scape's extensive library includes just about all the data entry and display functions you'll ever need, including money functions, fully definable borders, and orthogonal field movement (get the latest list by calling for more information). And modifying our functions or writing your own is easy. C-scape adjusts automatically for CGA, EGA, monochrome, and the Hercules Graphics Card Plus in RamFont mode, and optionally writes directly to video memory, so it's flexible *and* fast.

■ Bridges to Power

C-scape includes examples of how to bridge to other powerful tools such as **c-tree** and **db_VISTA.** You'll be integrating demos to dictionaries to file handlers and database managers in no time. You can even use C-scape to provide the screen design for AI applications, using packages that support calls to C.

■ Clean, Complete Documentation

C-scape's documentation is a clear example of how to write for programmers in a hurry. A short introduction uses helpful examples to explain the C-scape design. Each function is documented separately. An index makes reference easy, and a quick-reference card provides a synopsis of each function.

■ Source Code Included/Portable/No Royalties/No Runtime License

Providing source code at no additional cost gives you the freedom to modify existing functions without raising cost as a barrier. The source code includes all the low level routines you might need to port C-scape to an unsupported machine or compiler. Speaking of barriers, you pay *no royalties or runtime license fees,* either.

■ Toll Free Support and Bulletin Board

To make your job even easier, we provide technical support (toll free), and access to our 24-hour Bulletin Board.

■ Easy Prices/Risk-Free Terms

Try C-scape for 30 days. If you are not satisfied, return it for a refund. When you register, we send you source code. Order C-scape today, or call toll free for more information. See why some very major companies are standardizing on C-scape.

C-scape (Lattice/Microsoft/others)	**Only $199**
C-scape with Dan Bricklin's Demo Program	**$259**
NEW **C-scape with DB Demo and db_VISTA** (RAIMA)	**$425**

Please add $3 for first class shipping; Massachusetts orders add 5% sales tax.

Oakland Group, Inc.
675 Massachusetts Avenue, Cambridge, MA 02139-3309

MasterCard VISA **CALL NOW!**
800-233-3733
617-491-7311

CIRCLE 36 ON READER SERVICE CARD

PUBLIC DOMAIN SOFTWARE REVIEW

Hercules cards, fast disk copies, and even more on trojans

By Tim Parker

A program that arrived recently in my mail slot seems perfect for user groups and those involved in software distribution (legally, of course!). Jim Nech (10114 Torrington, Houston, Texas 77075, (713) 941-3100) has released a program called FastCopy into the shareware distribution scheme. FastCopy allows diskettes to be duplicated considerably faster than is possible with the standard DOS DISKCOPY and COPY programs, and rivals the speed of some of the commercial disk duplicators I have worked with.

FastCopy requires an IBM or compatible with 640K of RAM and allows quick and easy bulk formatting and copying of single or multiple source diskettes. One or two disk drives are required. Currently, FastCopy works with double-sided, 9-sector per track, 40 TPI formats only. This will probably be enhanced in a commercial version to be released soon.

A number of command options control the entire FastCopy operation. The source and destination diskettes are specified, with *ON* or *OFF* statements for four parameters: format forcing, format only, verify, and fast drives. If turned on, format forcing causes FastCopy to unconditionally format the destination diskettes. If left off, FastCopy checks the destination diskettes and performs a format for an initialized diskette only if one is required.

The format option contains one rather interesting anomaly. If the *format only* option is activated, FastCopy looks at the diskette with which the option was started, checks the layout of the diskette, and formats all subsequent diskettes that have identical layouts. In other words, if DOS has locked out some sectors on the start diskette because of bad sectors, those sectors will be locked out on every copy! This isn't really anything to complain about: the increase in speed more than warrants the trouble to get a defect-free start disk.

(For those who are interested, the FastCopy format routine copies the first track, which contains the file allocation table (FAT). It is not necessary to copy all of the other tracks, just format them. If FastCopy copies the FAT, it is understandable why the copies will have the same bad track information.)

The *verify* option, as expected, verifies the copies track-by-track. The final option, called *fast drives*, is made for machines with high-quality fast diskette drives that perform well under faster operations. Only experimentation will tell if this option should be used with your machine. (In testing, I found that my IBM PC-1 drives wouldn't work well, nor did a clone from Korea that used really cheap drives. A Compaq I rebuilt with Mitsubishi drives, however, practically zoomed along.)

The documentation accompanying the FastCopy program is several pages long, with brief instructions and a registration form for the program. The author requests a registration fee of $20. The documentation points out a fairly important point for FastCopy users: do not use the Control-Alt-Del sequence to interrupt FastCopy. The escape key should be used instead. I tried to crash the program several times by doing a reset while copying, and although the program recovered four times out of five with no problems, the fifth attempt trashed a floppy. I am not sure yet whether FastCopy caused the crash or if it was simply a coincidence, but the old adage to follow instructions is a good idea!

Although FastCopy may have limited appeal for many users it is a godsend to user groups. I also found it extremely useful recently when performing a full backup of all my master floppies. The program worked every time, on a wide variety of machines. This program is the solution for those with a floppy copy problem (is there a limerick in there somewhere?) and a welcome addition to the public domain.

As a Hercules graphics card user, I lament the fact that most software with graphics capabilities will not run without a CGA unless their authors have added the Hercules routines. A recent addition to my bulletin board should be of interest to all Hercules users.

SIMCGA, written by Chuck Guzis (153 N. Murphy Ave., Sunnyvale, Calif. 94086), is a series of routines that attempts to fool a Hercules card into believing it is CGA compatible. The simulation is full screen, with the proper aspect ratio.

The trick, apparently (let me warn you that I do not specialize in graphics routines), is that a CGA card uses two lines per character time, whereas the Hercules uses four. SIMCGA copies one line to a third displayed line to provide a filled-out image and employs three character lines. I'm totally lost now. After monitoring a hardware debugger board while the routine ran, it appears to me that INT 8H is used for this, which is mentioned in the single page documentation file.

The documentation lists several problems with the technique, some of which I found true and others that were not apparent during testing. The copy to the third line is necessarily real time, and some ghosting with a moving image results. Further, because of the aspect ratio differences (vertical pixels to horizontal pixels) between the CGA and Hercules card, the vertical image is scaled about 50% larger, so images are stretched somewhat because of the line doubling. This stretching was not a problem in my tests except where line-drawing techniques with thin lines were used.

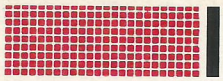
The CGA 40x25 text mode is not supported directly but is mapped onto a 80x25 mode. The only program I found that required this was an old BASIC game. Further, the documentation states that any program using the CGA's registers and starting directly at 03D4 will not work. I had no troubles with that in the brief testing I did.

The SIMCGA package is usually distributed as an ARC file composed of the single-page documentation file, a SIMCGA memory-resident program, and two programs to toggle between monochrome-only and CGA compatible modes. The resident program fits nicely into the AUTOEXEC program with the monochrome selector command following it, so the CGA-compatible routine can be executed from a batch file for any pro-

gram requiring it.

Testing the SIMCGA program showed that it is rather selective about the software it runs. BASICA worked fine, as did some games. Utilities and nongames seem to have a better chance of working, presumably because the games do direct memory addressing and strange register manipulations.

I also found that the SIMCGA package would not work on my 286- or 386-based Apricot machines with Hercules compatible graphics but would work on a clone running at 7.17 MHz. I suggest testing as the only real way to determine if SIMCGA is suitable.



In "Pirates, hacks, and trojans" (*COMPUTER LANGUAGE*, March 1987, pp. 75-78), I talked about preventing trojan programs (programs designed to damage your disk drives or information therein upon execution). I stated that "if DOS cannot access the drive, no damage can be done." While this is definitely true, the statement confused some readers. I appreciate the comments from several readers, in particular John Fowler and Jon Radel, both of whom took the time to write about some aspects of the subject I had not considered.

As mentioned, if DOS does not have access to the drive, it cannot do any damage. However, as several readers pointed out, a trojan program will not necessarily use DOS but can directly address the ports on a hard disk controller. This direct device addressing is not particularly difficult for a knowledgeable programmer, as the ports used for the drives are usually consistent. A command sent to the I/O address could bypass DOS, and perhaps even the BIOS, and perform format routines supported by the drive controllers.

It seems that most of the clone disk drive controllers that appear on the market at low cost are particularly susceptible to attack in this way. Indeed, some controllers provide a firmware format routine that can be activated with an I/O address. For IBM-compatible machines, which must by necessity support a number of particular address functions, a determined programmer can try the various addresses until he or she achieves some results (such as damage). For non-IBM machines, a hit-and-miss approach will accomplish the same thing.

It is impossible to protect against this sort of destructive attack. It has long been said that any devised defense can be broken, and in the case of hardware dependent functions this is more than true. (I spend a good part of my consulting life warning clients about possible security problems with their computer installations, and have seen this confirmed more times than I care to think about.) The casual user's only defense, as I discussed in my March column and in "Trojans revisited, CROBOTS, and ATC, (*COMPUTER LANGUAGE*, April 1987, pp. 91-93) is common sense, some care, and a little bit of wariness. Programs such as ASCII, which I discussed in April, that produce a dump of messages should be a first line of defense against such things.

Radel made a point about another comment in my March column concerning the possibility of damage to monitors, which I offhandedly said I didn't think could be damaged by a trojan. A perfect example of my error is the Hercules card, which can be destroyed by accident with a controller card and software performing a simple routine such as screen blanking. Hercules themselves warn purchasers against the problem: the imported Hercules-clone cards don't.

Indeed, some software (STSC's Statgrafix comes to mind immediately) warns the user that damage can result to the monitor if it is configured incorrectly during installation. Although I haven't investigated the cause of the problems much, I have checked with several service centers in my area, and they confirmed that they get a fair number of cases of monitor damage every year.

Complete C Programs in Half the Time, with *Instant-C*™

***Instant-C* helps you create a working, well-tested program *faster* than any other interactive C development system. Much faster than traditional compilers, linkers, and debuggers.**

Hard to believe? Here's how we do it. Because *Instant-C* is a high-performance interpreter there are **no compile or link delays.** Change your program, then test it immediately. No matter how large your program, the turnaround time is just seconds.

"Instant-C means instant gratification."—*PC Magazine*, **Editor's Choice** for best C interpreter. *10/29/85*

"Time after time, the Instant-C prompt was starting back just barely after pressing Enter."—*PC Tech Journal, 5/86*

Source-level debugging saves your time.

- set any number of conditional breakpoints in your program;
- stop execution from keyboard;
- single-step by source statement;
- examine and change variables or code, and continue execution;
- execute any statement or function directly for instant testing;
- display source code back-traces;
- source code animation;
- monitor data changes;
- full-screen and multi-screen support, even with non-standard graphics devices.

"The resulting debugging and testing capabilities are fantastic and the detailed trace/debug/display commands make it easy."—*The C Journal, 5/85*

Run-time checking stops your program as soon as errors occur, when bugs are easiest to understand and fix.

- pointer references checked for reasonableness;
- array indexes checked within declared bounds.

Not only does *Instant-C* help you quickly change, test, check and debug your code, but it runs your program faster than any other C interpreter. **Fifty to 500 times faster!** Fast enough for *real* programs, even fast enough for real-time programs.

"It is much faster than any of the other products mentioned and was the only one able to complete the standard SIEVE in a reasonable time. Clearly, this high speed allows much more complex problems to be attacked with Instant-C than with any of the other products discussed."—*Computer Language, 2/86*

Immediate feedback and more than 400 diagnostics makes *Instant-C* great for learning C. Full K&R and the ability to **link compiled object code and libraries** (Lattice and Microsoft) makes *Instant-C* compatible with your existing programs.

"When you get right down to it, I don't think there's a better way you could learn C."—*Programmer's Journal, 3/85*

"Clearly, Instant-C is the performance champion."—*PC Tech Journal, 5/86*

The bottom line for your business is increased productivity. The result for you is a job well done, and quickly.

"We sincerely feel that Instant-C can have a major positive impact on programmer productivity."—*Computer Language, 2/85*

"Instant-C by Rational Systems is a C programmer's dream."—*Micro/Systems Journal, 3/86*

Version 2 is available for MS-DOS and PC-DOS, and comes with a full 31 day **money back guarantee.** *Instant-C* is only $495. Order today! Call or write for full information.

Rational Systems, Inc.
P.O. Box 480
Natick, MA 01760
(617) 653-6194

CIRCLE 37 ON READER SERVICE CARD

Turbo Pascal Programmers

sophisticated software

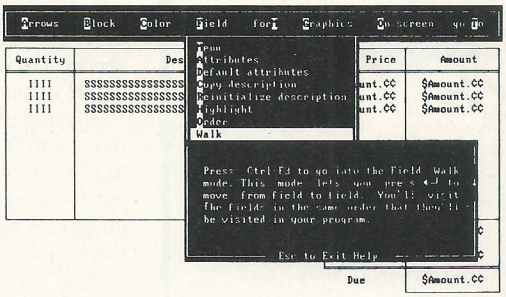
Presents

turbo magic™

The World's Best Code Generator!

turboMAGIC™ saves you time

MAGIC will generate all of the Turbo Pascal code you need for your input forms, help windows, pop up menus, and pull-down menu systems.

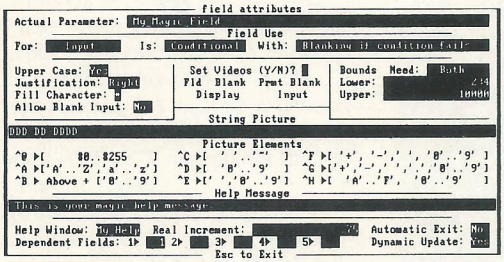


Sample PULL-DOWN MENU SYSTEM with on-line help

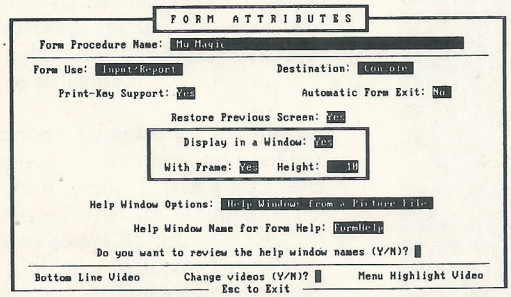
30 DAY MONEY BACK GUARANTEE!

turboMAGIC™ is easy to use

1. Paint the form you want with the powerful editor.
2. Select any special attributes that you want for fields or the form.
3. Let *turboMAGIC*™ do the rest.



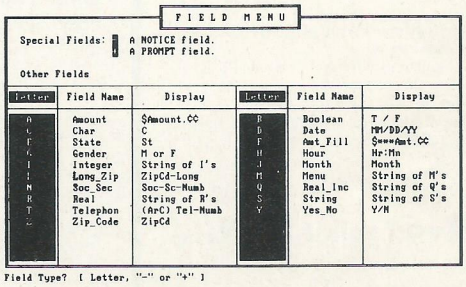
The FIELD ATTRIBUTES screen lets you describe a field.



The FORM ATTRIBUTES screen lets you describe a form.

turboMAGIC™ provides the ordinary... and a bit of the extraordinary!

- Forms that scroll in framed windows
- Forms up to 66 lines long using any colors and graphics characters
- Automatic updating of dependent fields
- Conditional-use fields (field and prompt can disappear if not needed)
- A help window per field or menu item
- Full-featured editing in fields
- Bomb-proof input handlers
- Support for printing forms, with video to font translation
- 19 Built-in field types available and you can add your own!



The FIELD MENU — an extensive set of types

A PROFESSIONAL OPINION: *"Fast automatic updating of dependent fields adds flair to your input screens. . . . turboMAGIC will be a blessing for programmers who would rather not write the user interface for every program."* Neil Rubenking, *PC Magazine*, Feb. 24, 1987

ORDER TODAY. $99

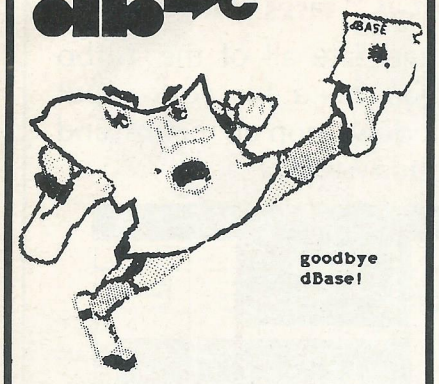
CALL 800-225-3165 or 205-342-7026

Requires 256K IBM and Turbo Pascal 3.X

Sophisticated Software, Inc.
6586 Old Shell Road
Mobile, Alabama 36608

CIRCLE 38 ON READER SERVICE CARD

dB→c



dBASE Programmers

are you ready?

dB→c

If you need it, and think you can handle it, dB2c is here now.

dB2c Offers:

- Complete implementation of dBASE III+ functions and commands in C source code.
- Interfaces for Lattice's dBC and Faircom's c-tree.
- Complete source code.
- Support for the following C Compilers: Microsoft, Lattice, Aztec, DeSmet, and Instant-C.
- The freedom and peace of mind that only source code can offer.
- Use of familiar syntax while learning C.
- Works on color, monochrome, and EGA monitors.
- Versions available for MS-DOS, Xenix, Unix, OS-9 and Concurrent DOS.

dB→c Toolkit $299.

DAVID J. MARSH

Call or Write:

SOFTWARE CONNECTION, INC.
POB 712, Ely, MN 55731
(218) 365-5097

CIRCLE 39 ON READER SERVICE CARD

One reader who took the time to write a complimentary letter (I like those!), asked how I test the software I review.

I test all software discussed in this column over a period of several weeks prior to writing the column. On some occasions, however, new software is sent to me or loaded on the bulletin board that I feel should get exposure quickly. In such cases I'll put the software through a shorter testing period. (An excellent example of such a case is SIMCGA.)

A complex program naturally requires more testing than a simple game. After I have assured myself that it is not a trojan (or simply a dumb program), I test it on several machines, in several configurations, and learn it as well as I can.

A side issue is whether some of the more esoteric boards will allow the software to function. I ran into some programs that simply refused to work with certain mono cards, and others that conflicted with expanded memory. Whenever possible I mention these, but please remember I have only a few machines, and not every board!

Some programs are latent trojans that do not effect their disasters until some time has passed. I encountered one program about two years ago that worked fine for almost a month, and the very day I sat down to write about it in this column, it trashed the disk! Examination of a copy made earlier revealed that it checked the system date for the 28th of the month, before unleashing itself!

In general, you can be fairly certain that if the program is in general distribution and you have not encountered any warnings about it on bulletin boards or from friends, it is probably a safe bet.

Finally, wrapping up with a last look at a program I mentioned in my April column, called NOTROJ, there has been quite a bit of activity regarding the program through the electronic mail systems. The NOTROJ program may be a very clever trojan after all!

At least two correspondents have mentioned hard disk erasures as a result of keeping NOTROJ resident, and several have relayed accounts of other such incidents. A caution to the user is strongly suggested. It appears, in all the reported cases, that NOTROJ only works when the hard disk is almost full, running at 70% or above of capacity. I have not had this occur on my test systems, and one of them is at 95% capacity. NOTROJ was left on that system for two days, with intermittent use, with no trouble.

Whether NOTROJ is a trojan or not I cannot definitively say. However, the intelligent user will take these warnings to heart, and use care when playing with the program. Backups should be mandatory of all systems that will use NOTROJ.

If indeed NOTROJ is a trojan, the author, although wreaking havoc on the users, has done a really clever job! Although I despise trojans and their authors, if NOTROJ is a trojan, it is the most elegant and well hidden trojan I have yet encountered. My hardware debuggers, interrupt trappers, and other normal trojan detectors let it slip right through.

As we have said here repeatedly, there is no way software can be tested to the absolute limit to ensure safety. Although I, and all other public domain authors try our best to avoid such occurences, it is inevitable that we get caught by a really clever program in the end. Users should always bear in mind the first law of public domain software: guilty until proven innocent!

Assemblers/Simulators/Compilers

Macro Assemblers/ Cross Assemblers for Series 3.0

Series 3 requires at least 256k of memory to run under MSDOS & CP/M86 and 54k TPA of memory to run CP/M80. Series 3 products have: Full Listing Control, Conditional Assembly & Built in Cross Reference. There is Unlimited Program Size, Number of Symbols and Macros. The Linkers output: Intel Hex, Extended Intel Hex, Tektronix Hex, and Motorola S19, S28, S37 formats.

Macro Assemblers/ Cross Assemblers for Series 4.0

Series 4 requires 512k of memory to run. These products have all the features of Series 3 plus: 32 Character Labels, Library and User Defined Sections. There is one Linker on Series 4 which outputs all the same formats as Series 3. Series 4 runs under MSDOS, UNIX, VMS and ULTRIX.

Simulator-Debuggers

The Simulator-Debuggers run with 256k of memory. The Simulator has 16 Breakpoints with optional Counter Field. The Symbol Table is limited only by the amount of memory. Buffers of 256 bytes may be opened for I/O ports. The Simulators are available for MSDOS, UNIX, VMS and ULTRIX.

C Compilers

The C Compilers support in-line assembly language and ROMable code, and includes the following: Macro Processor, full Floating Point support, complete Assembler, Linker, and Libraries.

	Macro Assemblers Cross Assemblers			Simulator-Debuggers		C Compilers	
	Series 3	Series 4					
Software Product	CP/M 80 CP/M 86 MSDOS	MSDOS AT&T UNIX	VMS Zeus ULTRIX	MSDOS AT&T UNIX	VMS Zeus ULTRIX	MSDOS AT&T UNIX	VMS Zeus ULTRIX
Super 8		199.50	995.00	149.50	499.50		
Z-8	199.50	199.50	995.00	149.50	499.50		
Z-80	199.50	199.50	995.00	149.50	499.50	500.00	2000.00
Z-280		299.50	1250.00				
Z-8000	299.50	299.50	1250.00				
1802	199.50						
6301	199.50	199.50	995.00	149.50	499.50		
64180	199.50	199.50	995.00			500.00	2000.00
6501	199.50	199.50	995.00				
6502	199.50	199.50	995.00				
65c02	199.50	199.50	995.00				
65c816		299.50	1250.00				
6800,2,8	199.50	199.50	995.00	149.50	499.50		
6801,3	199.50	199.50	995.00	149.50	499.50		
6804	199.50	199.50	995.00				
6805	199.50	199.50	995.00				
6809	199.50	199.50	995.00	149.50	499.50		
68c11	199.50	199.50	995.00				
68000,8,10	299.50	299.50	1250.00				
68020		399.50	1500.00			700.00	2500.00
8400/c00	199.50	199.50	995.00				
80452		199.50	995.00				
8044/51	199.50	199.50	995.00	149.50	499.50		
80515		199.50	995.00				
8080	199.50	199.50	995.00	149.50	499.50		
8085	199.50	199.50	995.00	149.50	499.50		
8086/88	99.50	99.50	1250.00				
8096	199.50	199.50	1250.00				
80186/286	199.50	199.50	1250.00				
83c351		199.50	995.00				
8748	199.50	199.50	995.00	149.50	499.50		
V20/30	199.50	199.50	1250.00				
Cops 400	199.50						
F8/3870	199.50						
NCR32	399.50						
NEC7500	199.50						
NSC800	199.50	199.50	995.00	149.50	499.50	500.00	2000.00
32000	399.50						

More formats available

Name _____
(Please Print)

Company _____

Address _____

City _____ State _____ Zip _____

Phone _____

Product _____ Operating System _____

Series _____ Amount $ _____

Shipping $ _____ Total $ _____

Signature _____

Check ☐ COD (U.S. Only) ☐

MC/VISA # _____

Expiration Date _____

Educational discount available.

To order, call toll free in U.S. (including HI, PR and VI):

1 800 843-8144

In Colorado: (303) 369-5001

For more details, ask for a free brochure.

(Shipping is $20.00 per unit for overseas orders. Toll Free number does not apply to overseas. 2500AD pays COD charges.)

2500ADSOFTWARE INC.

17200 East Ohio Drive
Aurora, Colorado 80017
(303) 369-5001

CIRCLE 27 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

UI Programmer - Quickly generate dBASE User Interfaces, prototypes. Use supplied templates or create own. Pop-up help, bounce bar menus, screen forms. II, III, FoxBASE+, Quicksilver, Clipper PC $ 295

AI-Expert System Dev't

Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
SQL Dev't Package	MS	$ 229
Auto-Intelligence	PC	$ 749
Experteach - Powerful, samples	PC	$ 349
Exsys	PC	$ 309
Runtime System	PC	$ 469
Insight 1	MS	$ 75
Insight 2+	MS	$ 379
Intelligence/Compiler	PC	$ 749
Texas Instruments:		
PC Easy	PC	$ 435
Personal Consultant Plus	PC	$2589
Personal Consultant Runtime	PC	$ 85
Turbo Expert-Startup (400 rules)	PC	$ 129
Corporate (4000 rules)	PC	$ 359

AI-Lisp

Microsoft MuLisp 85	MS	$ 159
PC Scheme LISP - by TI	PC	$ 85
TransLISP - learn fast	MS	$ 89
TransLISP PLUS		
Optional Unlimited Runtime	MS	$ 150
PLUS for MSDOS	MS	$ 179

Others: IQ LISP ($239), IQC LISP ($269)

AI Prolog

APT - **A**ctive **P**rolog **T**utor - build applications interactively	PC	$ 65
ARITY Prolog - full, 4 Meg		
Interpreter - debug, C, ASM	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
Standard Prolog	MS	$ 77
MacProlog Complete	MAC	$ 269
MicroProlog - Prof. Entry Lev.	MS	$ 85
MicroProlog Professional Compiler/Interpreter	MS	$ 439
MPROLOG P550	PC	$ 175
Prolog-86 - Learn Fast	MS	$ 89
Prolog-86 Plus - Develop	MS	$ 229
TURBO PROLOG by Borland	PC	$ 69
Turbo Prolog Toolbox	PC	$ 69

Basic

Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 89
Basic Windows by Syscom	PC	$ 95
BetterBASIC	PC	$ 129
8087 Math Support	PC	$ 75
Run-time Module	PC	$ 169
Exim Toolkit - full	PC	$ 39
Finally - by Komputerwerks	PC	$ 85

FEATURES

Personal COBOL by MicroFocus- Develop, test, debug, execute ANSI '74 code. Full-screen editor, syntax checker, Animator, forms/ screen generator, help. Compatible with Level II. PC $ 169

Turbo C by Borland. ANSI Compiler supports 6 models including tiny and huge, has floating point, interactive editor, Make. Speed development. PC $ 69

Compare Products Use Product Specialists

The Programmer's Shop is much more than the supplier of the largest selection of programmer's software. Trained programming consultants will answer your questions. Ask "product specialists" about Cross Assembler, Translators, Debuggers, or compilers.

Our Services:

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS - 7 PM to 7 AM 617-740-2611
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

Basic Cont.

QuickBASIC	PC	$ **69**
Turbo BASIC - by Borland	PC	$ 69

Cobol

Microfocus Professional Cobol	PC	$2295
VS Workbench	PC	$3379
Microsoft COBOL	MS	$ 439
Microsoft Cobol Tools	PC	$ 209
Realia - very fast	MS	$ 819
Ryan McFarland COBOL	MS	Call
COBOL-8X	MS	Call
Screenplay - screen mgmt.	PC	$ 129

C Libraries-Communications

Asynch by Blaise	PC	$ 125
Essential Comm Library	PC	$ 135
With Debugger	PC	$ 195
Greenleaf Comm Library	PC	$ 129
Multi-Comm - add multitasking	PC	$ 149

dBASE Language

Clipper compiler	PC	Call
dBASE II	MS	$ 329
dBase III Plus	PC	$ 429
dBASE III LanPack	PC	$ 649
DBXL Interpreter	PC	$ 139
FoxBASE+ - single user	MS	$ 349
QuickSilver by Word Tech	PC	$ 499

dBASE Support

dBase Tools for C	PC	$ 65
dBrief with Brief	PC	Call
dBC ISAM by Lattice	MS	Call
dBx Translator to C	MS	$ 319
dFlow - flowchart, xref	MS	Call
Documentor - dFlow superset	MS	Call
Genifer by Bytel-code generator	MS	$ 299
QuickCode III Plus	MS	$ 249

Editors for Programming

BRIEF Programmer's Editor	PC	Call
EMACS by UniPress Source: $929		$ 299
Epsilon - like EMACS, full C-like language for macros.	PC	$ 149
KEDIT - like XEDIT	PC	$ 99
Micro Focus Micro/SPF	PC	$ 139
PC/EDT - macros	PC	$ 229
PC/VI - by Custom Software	MS	$ 109
Personal REXX	PC	$ 99
PMATE - power, multitask	PC	$ 119
SPF/PC - fast, virtual memory	PC	$ 139
Vedit	MS	$ 107
Vedit PLUS	MS	$ 139

RECENT DISCOVERY

COBOL I/Q - Ad hoc queries, customizable reports and graphs from Micro Focus, RM, Microsoft, mbp COBOL, Lotus, dBASE, Btrieve, ASCII files. PC $539

C Language-Compilers

AZTEC C86 - Commercial	PC	$499
C86 PLUS - by CI	MS	Call
Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev'rs Kit	PC	$ 77
Datalight Optimum-C	MS	$ 99
with Light Tools by Blaise	PC	$168
Lattice C - from Lattice	MS	$275
Let's C Combo Pack	PC	$ 99
Let's C	PC	$ 59
Microsoft C 4.0- Codeview	MS	**$275**
Rex - C/86 by Systems & Software - standalone ROM	MS	$695
Uniware 68000/10/20 Cross Compiler by SDS	MS	Call
Wizard C	MS	$299
ROM Development Package	MS	$259

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$229
C Trainer - by Catalytix	PC	$ 89
INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs	MS	$379
Interactive C by IMPACC Assoc.	PC	$209
Run/C Professional	MS	$159
Run/C Lite	MS	$ 89

C Libraries-General

Blackstar C Function Library	PC	$ 79
C Essentials - 200 functions	PC	$ 75
C Function Library	MS	$109
C Tools Plus (1 & 2) - Blaise	PC	$125
C Utilities by Essential	PC	$129
C Worthy Library - Complete, machine independent	MS	$249
Entelekon C Function Library	PC	$119
Entelekon Superfonts for C	PC	$ 45
Greenleaf Functions-portable, ASM		$139
LIGHT TOOLS by Blaise	PC	$ 69

C Libraries-Files

FILES: C Index by Trio - full B+ Tree, vary length field, multi compiler		
/File is object only	MS	$ 89
/Plus is full source	MS	$319
CBTREE - Source, no royalties	MS	$ 99
CTree by Faircom - no royalties	MS	$319
rtree - report generation	PC	$239
dbQUERY - ad Loc, SQL - based	MS	$129
dbVISTA - full indexing, plus optional record types, pointers, Network.		
Object only - MS C, LAT, C86		$129
Source - Single user	MS	$389
Source - **Multiuser**	MS	$799
dBx - translator	MS	$299
w/source to library	MS	$459

FEATURE

Star Sapphire by Sapiens Software. Common LISP compiles/translates to C. Features 8M virtual memory workspace, lexical & dynamic scoping. C Library source, graphics. Flavors, unlimited multidimensional arrays. MS $ 459

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

CIRCLE 41 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

"Favorite Libraries"

These popular packages provide pre-written code, increasing your productivity. Professionally-developed routines for graphics, windows, communications, file handling, and more. Multi-Language support.DON'T reinvent the wheel. Call one of our specialists TODAY.

Order before June 30, 1987 and mention this ad for these SPECIAL PRICES:

	List	Normal	SPECIAL
Btrieve-single user	$245	$199	**$169**
Network version	$595	$499	**$419**
Greenleaf Comm Library	$185	$129	**$109**
Greenleaf Data Windows with Source	$225 $450	$159 $299	**$129 $249**
Greenleaf Functions	$185	$139	**$109**
Halo	$300	$209	**$179**
Microsoft Dev't Pkg.	$595	$395	**$349**
Panel	$295	$215	**$179**

C Support-Systems

Advantage C++	PC	Call
Basic-C Library by C Source	PC	$139
C Sharp - **realtime,** tasks.	PC	$600
C ToolSet - DIFF, xref, source	MS	$ 95
The HAMMER by OES Systems	PC	$139
Lattice Text Utilities	MS	$ 89
Multi-C - multitasking	PC	$149
PC LINT-Checker. Amiga $89	MS	$ 99
Quickshell - script compiler	PC	$349
Pfantasy Pac - by Phoenix	PC	$849
PforCe++	PC	$279
Pre-C - Lint-Like	MS	$155
Programmer's Extender	MAC	$ 79
Sapiens V8 - 8M workspace	PC	$269
SECURITY LIB-add encrypt to MS C, C86 programs. Source $229	PC	$115

C-Screens, Windows, Graphics

C Power Windows by Entelekon	PC	$109
dBASE Graphics for C	PC	$ 69
C-Scape - capture Dan Bricklin	PC	$179
Curses by Lattice	PC	$ 89
ESSENTIAL GRAPHICS - fast	PC	$195
GraphiC - new color version	PC	$285
Greenleaf Data Windows	PC	$159
w/source	PC	$299
Multi-Windows - use w/ Multi-C	PC	$295
Screen Ace Form Master	PC	$195
Vitamin C - screen I/O	PC	$159
Windows for C - fast	PC	$189
Windows for Data - validation	PC	$319
View Manager - by Blaise	PC	$189
ZView - screen generator	MS	$169

Debuggers

386 Debug - by Phar Lap	PC	$129
Breakout - by Essential	PC	$ 89
CODESMITH - visual	PC	$ 99
C SPRITE - data structures	PC	$129
DSD87 - by Soft Advances	PC	$ 79
Periscope I - List: $345	PC	Call
Periscope II - List: $175	PC	Call
Periscope II-X - List: $145	PC	Call
Pfix-86 Plus - by Phoenix	PC	$229
Showcase - test software	PC	$135
SoftProbe II - embedded systems	PC	$695

FEATURE

TimeSlicer - Multitasking, realtime processing, intertask communication. Resource management, task scheduling, preemption control. MS, Lattice C, or Assembler. No Royalties. PC $265

Fortran & Supporting

50:More FORTRAN	PC	$ 99
ACS Time Series	MS	$399
Forlib+ by Alpha	MS	$ 59
MACFortran by Microsoft	MAC	$229
MS Fortran - 4.0, full 77'	MS	$279
No Limit - Fortran Scientific	PC	$115
PC-Fortran Tools - xref, pprint	PC	$179
RM/Fortran	MS	Call
Scientific Subroutines - Matrix	MS	$139

Multilanguage Support

BTRIEVE ISAM	MS	$185
BTRIEVE/N-multiuser	MS	$455
Flash-Up Windows	PC	$ 79
GSS Graphics Dev't Toolkit	PC	$375
HALO Graphics	PC	$209
Development Package	MS	$395
Informix - by RDS	PC	$639
Informix 4GL-application builder	PC	$789
Informix SQL - ANSI standard	PC	$639
Opt Tech Sort - sort, merge	MS	$ 99
PANEL -	MS	$215
Pfinish - by Phoenix	MS	$229
PolyLibrarian by Polytron	MS	$ 79
PolyBoost - speed I/O, keyboard	PC	$ 69
PVCS Corporate-source control	MS	$319
QMake by Quilt Co.	MS	$ 85
Report Option - for Xtrieve	MS	$109
Screen Sculptor	PC	$ 95
SRMS - source control	MS	$109
Synergy - create user interfaces	MS	$375
VXM - multi-env. link	MS	$195
Xtrieve - organize database	MS	$199
ZAP Communications - VT 100	PC	$ 89

Pascal and Supporting

ALICE - learn Pascal	PC	$ 59
Exec - Chain Programs	MS	$ 79
MetaWINDOWS-graphics toolkit bit-mapped, fast	PC	$115
MetaWINDOWS PLUS	PC	$185
Microsoft PASCAL - faster	MS	$189
Programmer's Extender	MAC	$ 79
Pascal Pac with Tidy - formatter, utilities	PC	$ 69
Pascal Tools PLUS	PC	$125
Pascal 2 - by Oregon Software, tight, fast	MS	$329
TurboHALO - 150 routines	PC	$ 99

DataBase & File Management

DataFlex by Data Access	PC	$ 899
Dataflex multiuser	PC	$1149

RECENT DISCOVERY

NET-TOOLS - Access NETBIOS-compatible network systems from Microsoft C, Pascal, FORTRAN, Assembler, Lattice C. Full Source, No Royalties. PC $ 129

Other Languages

APL*PLUS/PC	PC	$ 429
CCS Mumps - Singleuser	PC	$ 50
CCS Mumps - Multiuser	PC	$ 359
MasterForth - Forth '83 MAC or	PC	$ 109
Microsoft MASM	MS	$ 98
Modula-2 - by Pecan	MS	$ 79
Modula-2/86 by Logitech	PC	$ 62
Pasm - by Phoenix	MS	$ 115
PC Forth+ - by Lab Micro	PC	$ 199
Smalltalk/V	MS	$ 85
SNOBOL4+ - great for strings	MS	$ 80
UR/Forth	MS	$ 279

Xenix/Unix

Basic - by Microsoft	$ 239
C-Terp by Gimpel Software	$ 379
Cobol - by Microsoft	$ 639
Cobol Tools - by Microsoft	$ 319
Fortran or Pascal - by Microsoft	$ 439
MicroFocus Lev. II Compact COBOL	$ 795
Panel	$ 535
RM/Cobol	Call
RM/Fortran	Call
Xenix Complete System	$1049

Other Products

386 Assembler/Linker	PC	$ 389
ASMLIB - 170+ routines	PC	$ 129
asmTREE - B+ tree file mgmt.	PC	$ 339
Compact Source Print	PC	$ 45
Dan Bricklin's Demo Program	PC	$ 59
Help/Control - on line help	PC	$ 99
Interactive Easyflow-HavenTree	PC	$ 125
Link & Locate - tools to work with Intel and Tektronix projects.	MS	$ 329
LMK - like UNIX make	MS	$ 139
Microsoft Windows Software Development Kit	PC	$ 319
MKS Toolkit - Unix, vi, awk	PC	$ 99
Norton Commander	PC	$ 55
Numerical Analyst by Magus	PC	$ 269
PDisk - cache, tree	PC	$ 99
PLink - 86 PLUS - overlays	MS	$ 319
PMaker - by Phoenix	PC	$ 79
Polymake by Polytron	MS	$ 119
PolyShell by Polytron	MS	$ 119
PolyXREF by Polytron	PC	$ 99
Sapiens V8 - 8M virtual mgr.	PC	$ 300
Taskview - by Sunny Hill Software, ten tasks	PC	$ 55
Tom Rettig's Library - dBASE	PC	$ 89
Tree Diagrammer	PC	$ 49
Visible Computer: 8088	PC	$ 65

Note: Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item. All prices subject to change without notice.

Call for a catalog, literature, advice and service you can trust

800-421-8006

VISA HOURS MasterCard

8:30 AM - 8:00 PM EST.

THE PROGRAMMER'S SHOP™

Your complete source for software, services and answers

5-L Pond Park Road, Hingham, MA 02043

Mass: 800-442-8070 or 617-740-2510 4/87

"I have been pleased with your catalog selection, the knowledge of your telephone answering staff, and the promptness of your service. I have every wish to become a continuing customer of The Programmer's Shop."

Carl C. Rollo

CIRCLE 42 ON READER SERVICE CARD

C PROGRAMMERS!

db_VISTA™: fast C Database now offers SQL-based Query

Performance . . .

High-speed data access and low data redundancy . . . just two benefits of using Raima's network model DBMS, db_VISTA. Combine these benefits with those of C—speed, portability, efficiency and you begin to understand db_VISTA's real measure . . . performance.

fast . . .

It's fast — almost 3 times faster than a leading competitor. Fast access that comes from the unique combination of the B-tree indexing method and the "network model" or direct "set" relationships between records. A winning combination for fast performance.

new db_QUERY™ . . .

Add our new C-linkable, SQL-based, ad hoc query and report-writing companion product to provide a simple relational view of **db_VISTA**'s complex network model database. Without compromising speed. Ask for **db_QUERY**!

portable . . .

db_VISTA and **db_QUERY** operate on most popular computers and operating systems like MS-DOS, UNIX, and VMS. You can write applications for micros, minis, or even mainframes, allowing the same C applications to run under MS-DOS, UNIX, and VAX VMS.

multi-user . . .

db_VISTA has multi-user capability and supports simultaneous users in either multi-tasking or local area networking environments.

here's how . . .

Design your database and compile your schema file with the database definition language processor. Develop application programs, making calls to **db_VISTA**'s C functions. Edit and review your database using the Interactive Database Access utility. Compile and link your C program with the **db_VISTA** run-time library, and your application is ready to run.

source code . . .

We make our entire C Source Code available so you can optimize performance or port to new environments yourself.

efficient . . .

db_VISTA uses space efficiently. It lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

royalty free . . .

Whether you're developing applications for yourself or for thousands, you pay for **db_VISTA** or **db_QUERY** only once. No extra run-time charges.

free support . . .

Raima's software includes free telephone support and software updates for 60 days. Technical support personnel are available to answer questions about our software or yours.

order now . . .

Put db_VISTA to work in your application program. Ordering is easy — simply call toll-free. We'll answer your technical questions and get you started. Call today.

Training Classes are available! Ask about it!

BASICS, ADVANCED, INTERNALS
June 15-19 Seattle, WA USA
Call for additional dates

	db_VISTA	db_QUERY
Single-user	$ 195	$ 195
Single-user w/Source	$ 495	$ 495
Multi-user	$ 495	$ 495
Multi-user w/Source	$ 990	$ 990
NEW:		
VAX Multi-user	$ 990	$ 990
VAX Multi-user w/Source	$1980	$1980

MasterCard VISA

30 day money-back guarantee!

db_VISTA™

Features

- **Multi-user** support allows flexibility to run on local area networks
- **File structure** is based on the B-tree indexing method
- **Transaction processing** assures multi-user consistency
- **File locking** support provides read and write locks
- **SQL-based db_QUERY** is linkable
- **File transfer** utilities included for ASCII, dBASE optional
- **Royalty-free** run-time distribution
- **Source Code** available
- **Data Definition Language** for specifying the content and organization of your files
- **Interactive database access** utility
- **Database consistency check** utility
- **Key file build** utility
- **ASCII file import and export** utility

File Management Record and File Sizes

- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- Maximum file size limited only by available disk storage
- Maximum of 256 index and data files
- Key length maximum 246 bytes
- No limit on number of key fields per record
- No limit on maximum number of fields per record

Operating System & Compiler Support

- **Operating systems:** MS-DOS, PC-DOS, UNIX, XENIX, UNOS, ULTRIX, Microport, VMS
- **C compilers:** Lattice, Microsoft, IBM, DeSmet, Aztec, Computer Innovations, Turbo C, XENIX and UNIX

CL 67

what others say . . .

"If you are looking for a sophisticated C programmer's database, **db_VISTA** is it. Raima's customer support and documentation is excellent. Source code availability and a royalty-free run-time is a big plus."

Dave Schmitt, President
Lattice, Inc.

"**db_VISTA** has proved to be an all-around high performer in terms of fast execution, flexibility and portability, and has undoubtedly saved us much time and development effort."

John Adelus, Hewlett-Packard Ltd.
Office Productivity Division

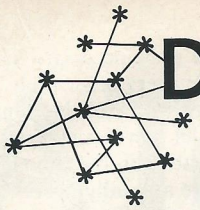
RAIMA™ CORPORATION

CIRCLE 43 ON READER SERVICE CARD

call toll-free today!

1 (800) db-RAIMA
(that's 1-800-327-2462)

3055 - 112th N.E., Bellevue, WA 98004 USA (206) 828-4636 Telex: 6503018237 MCIUW



DESIGNING WITH DATA BASES

Data base integrity

By Matthew Rapaport

We begin by taking up the subject of defined fields (also known as computed, or virtual fields). As their name implies, such fields are not data stored in the data base but rather representations derived from one or more attributes of stored data associated with some predicate operation (usually addition, multiplication, etc.).

Since these fields are normally required for some output, most systems provide for them as part of their query or screen languages. Like other characteristics of data, however, when computed fields are definable in the dictionary, the environment as a whole comes to have more flexibility. This is because the computed field not only can be named in queries but can itself be an object of another computation. Furthermore, it is not necessary for the programmer to know *how* to compute the field when designing a new report or screen. Programmatically, the field can be treated much as stored data.

Dunn & Bradstreet Corp.'s NOMAD2 and Microrim's R:base System V represent the kinds of defined fields typically under control of the data dictionary. R:base System V allows only two or more columns associated with an arithmetic predicate, and there can be no more than 10 computed columns in a table. NOMAD2 allows for such computations and for the *EXTRACT* type, a formulae in one table that references a foreign key (I'll discuss foreign and primary keys later) in its own tuple (row) and uses it to obtain some attribute value in a second table whose primary key has the same domain and value as the foreign key (Figure 1).

For example, using the JOB.NUMBER in the EMPLOYEE relation (file), one can define a field in EMPLOYEE called JOB.TITLE. This definition instructs the system to use the JOB.NUMBER in EMPLOYEE, match it to the same JOB.NUMBER in the JOB.TABLE, and then associate the JOB.TITLE thus found in the JOB.TABLE with the field named JOB.TITLE in EMPLOYEE. In this way, it looks to the application program like JOB.TITLE is actually stored in the EMPLOYEE table, when in fact it is being extracted from the JOB.TABLE as each record is processed. Unlike R:base System V, NOMAD:2 has no limit on the number of defined fields. Not only may entire tables be composed of them (though they cannot be keys), but the *EXTRACT* type defines may reference defined fields in other tables!

Cosmos' Revelation dictionary excels in its defined field capabilities. Both of the defines already mentioned are supported, but Revelation's capabilities here extend to almost anything that can conceivably be programmed in Revelation, including the opening of other tables and the processing of multiple records! For example, a given virtual field might represent the sum of an attribute value across many records in another relation.

This ability to embody in one field the result of executing numerous programs, opening other tables, and more is found only in Revelation and the Pick operating system from Pick Systems. Such potential is used sparingly, for there is enormous overhead involved in all that processing. Defined attributes can be very efficient as well.

Using them where appropriate generally speeds record inserts and updates and saves disk space. By contrast, they cause processing (queries or procedural access) to slow down because more CPU resources are required (note that a disk I/O must take place regardless of field type, as the tuple that contains the defined field must be retrieved before the field's value can be computed).

Integrity issues can be divided into four major categories. Category I embraces the notions of entity and referential integrity. This area deals most with data dictionary services, though many products provide what support they have in this category outside of the dictionary.

Category II includes those DBMS services which ensure that related, multi-transaction update sequences occur either as a whole group or not at all. The classic example is that of a banking transaction in which money is shifted from one account to another. One transaction updates the first account by subtracting some amount. A second transaction updates the second account by adding a like amount to it. If something happens and the second transaction fails, money will have been lost! What we want here is some way to prevent the first transaction from being irrevocably committed until we can be sure that the second will succeed. What we need is some way to write the transactions but flag them as unofficial until some procedure determines that both transactions have succeeded.

Another aspect of transaction integrity involves the utilities provided to recover from a situation where the process fails (possibly because of a power failure) midway through the sequence of transactions. This ability to recover is accomplished through a process called transaction logging. Transaction logging is essential in large-scale data processing, where all-or-nothing sequences may involve a dozen or more logical tables. Most of the larger systems, such as Sincom's Supra, Unify Corp.'s UNIFY, and Relational Database Systems Inc.'s Informix, feature these services. By contrast, the most popular small systems, such as Ashton-Tate's dBASE III+, R:base System V, Ansa Software Corp.'s Paradox, and Revelation, provide no transaction logging services whatsoever.

It is informative to note that the typical Management Information Systems application does not require extensive ability in this area. Unlike data processing, MIS updates do not often have consequences in multiple tables. Transaction logging would still be useful to MIS because it makes disaster recovery more straightforward, but it is not as critical a feature as it is in data processing.

Category III issues do not arise on single-user microcomputers. They arise only in connection with simultaneous update to a data base by several users on larger micros, minis, mainframes, and LANs. Concurrency integrity is the province of the scheduler on large computers and procedurally coded LOCKS in smaller ones such that one set of transactions is prevented from corrupting the otherwise valid transactions of another user. This

problem of interference comes about because the DBMS must handle parts of requests from many users before completing a logical group of transactions for each requester. This category of concerns works to ensure that transactions are executed in such a way that no integrity violations occur and that locking at various levels does not bring the entire DBMS to a halt.

Category IV is that of DBMS security: ensuring that no unauthorized access or update takes place. Of all the integrity types, security is the least connected to the basic elements of the DBMS.

A large body of theory surrounds Category II and III issues (References 1, 2, and 4), and a good deal of practical experience surrounds the issues of Category IV. We will discuss Categories II, III, and IV in another column, and focus now on the first class of integrity issues.

If two distinct tuples of a relation were permitted to have the same primary key, there would be no way to identify one from the other. As far as the DBMS is concerned, the same record would exist in two different forms, possibly with conflicting data values. The possibility of ensuring data integrity and consistency (consistency being a special case of integrity where two or more values are required to be in agreement based on some comparison) therefore depends upon the integrity of the primary key. Specifically:

1. The key, taken as a whole must be unique within a given relation.
2. All parts of the key must be necessary to ensure that rule 1 holds. This means that no redundancies should exist among the parts of a multipart key.
3. A corollary of the first two rules is that no part of the primary key can ever be null. If it could, then one of the first two rules would fail.

These rules, taken together, are called the Entity Integrity Rule. It is supposed to reflect the real universe being modeled in that no two objects in the real world are identical in every respect. There is always some way to distinguish between two real things or it would make no sense for us to say they were different. Since a tuple represents some thing in the real world, it too must be uniquely specifiable in some way. It is quite possible for every attribute of the tuple to be necessary in distinguishing it from every other tuple (though in practice this is rare). Nothing in the relational model prevents every attribute from participating in the primary key, provided they are all really necessary to making the tuple unique.

A tuple may have several attributes or attribute combinations that might be primary keys. If this is the case, one of these attributes or attribute combinations is designated the primary key, while the others are alternate keys. For example, customer number and customer name may both happen to be unique. Data in relational tables may be accessed with or without the use of a primary key, but access through a key is the only way to guarantee that at most one tuple will be found. For this reason, these keys are the real foundation of the relational algebra.

As a rule, the attributes chosen as the primary key should be stable and not subject to updates. This goal is sometimes difficult to achieve if the attributes of the key have some meaning of their own that may be subject to change. For example, a part type number whose first two digits represent a place of origin would have to be changed if production of that part were moved to another fac-

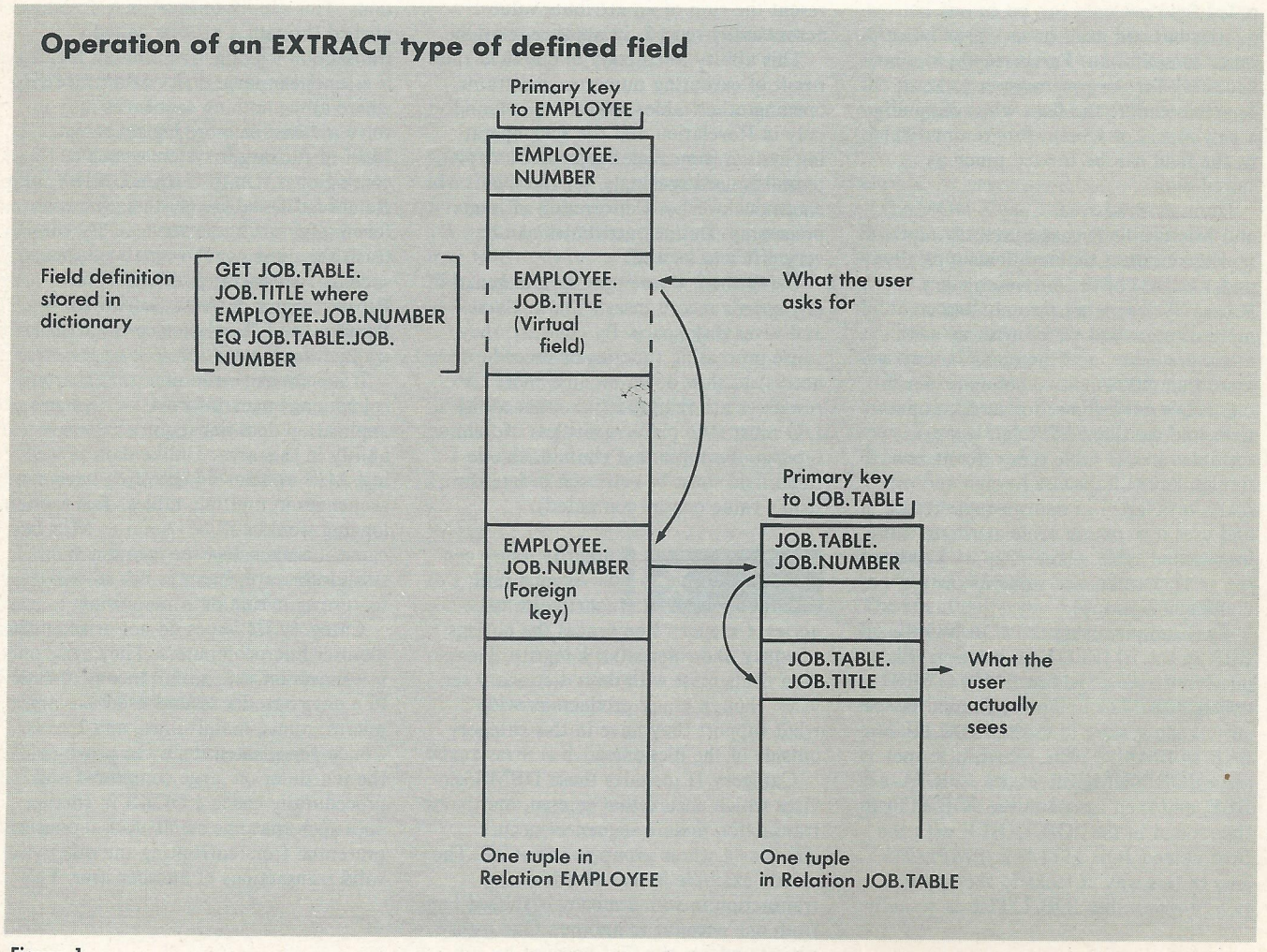


Figure 1.

Software development guides

An Investment in Your Professional Future...

Use them FREE for 15 days!

New!

APPLIED C

By **Strawberry Software, Inc.,** *edited by* **Bonnie Derman**

Let experts from Strawberry Software, Phoenix, Lattice, and Greenleaf show you how to get the most out of the C language. Fully covered are how to write program specifications...considerations of user interface design, style, portability, and modularity...data types, parsing, and indexing...and much more. **272 pp., illus., $37.95**

LISP

The Language of Artificial Intelligence

By **A. A. Berk**

Practical examples and careful explanations help you master the complexities of such concepts as recursion and data structures in this informative guide to LISP. BASIC is used as a reference point throughout. **168 pp., $28.95**

ARTIFICIAL INTELLIGENCE

An Applications-Oriented Approach

By **Daniel Schutzer**

Start putting AI to work for you with this first guide to current applications of expert systems, machine perception, and natural language processing. You'll find out how to use AI to solve a wide range of problems in business, industry, science, medicine, and defense. **224 pp., illus., $39.95**

New!

EMBEDDED PROGRAMMING IN ADA

By **Theodore Frederick Elbert III**

Covering virtually every aspect of the ADA language, this guide shows you how to implement both basic and advanced programming techniques. You'll get detailed information on packages, generic units, input-output, tasks, other language features, and more. **512 pp., illus., $45.95**

HANDBOOK OF SOFTWARE QUALITY ASSURANCE

Edited by **G. Gordon Schulmeyer** *and* **James I. McManus**

Use this handbook to create programs that meet or exceed performance specifications every time! It explains proven, state-of-the-art testing, reliability, and statistical procedures for assuring software quality. The uses of inspections, the Pareto principle, software configuration management, and more are detailed. **528 pp., 118 illus., $32.95**

SOFTWARE SYSTEM TESTING AND QUALITY ASSURANCE

By **Boris Beizer**

Using nontechnical, easy-to-understand language, this comprehensive guide covers everything from unit to system testing. It explains new and effective techniques for security testing, recovery testing, configuration testing, and performance testing. **384 pp., illus., $36.95**

HANDBOOK OF SOFTWARE ENGINEERING

Edited by **Charles R. Vick** *and* **C.V. Ramamoorthy**

Emphasizing quality assurance, this useful handbook shows how to develop and maintain software for micro- and minicomputers, array machines, distributed systems, and support systems. "A reader's digest of software knowledge for the budding and the practicing computer professional."—*IEEE Software.* **720 pp., 285 illus., $66.95**

Mail coupon today for your FREE-EXAMINATION copies!

VAN NOSTRAND REINHOLD
Mail Order Department
P.O. Box 668
Florence, KY 41042-9979

YES! Please send me the book(s) checked below for 15 days' FREE EXAMINATION. At the end of that time, I will send the purchase price plus local sales tax and a small shipping/handling charge, or simply return the book(s) and OWE NOTHING.

___ 28217-6 Applied C $37.95
___ 20974-6 LISP $28.95
___ 28034-3 Artificial Intelligence $39.95
___ 22350-1 Embedded Programming in ADA $45.95
___ 28015-7 Handbook of Software Quality Assurance $32.95
___ 21306-9 Software System Testing and Quality Assurance $36.95
___ 26251-5 Handbook of Software Engineering $66.95

Orders totalling over $100.00—Please enclose a check, money order, purchase order, or credit card information. Payment in advance saves you shipping/handling costs and you'll receive a full refund if not completely satisfied for any reason. Local sales tax must be included.

☐ **SAVE MONEY!** Check here if enclosing payment with order and publisher pays shipping/handling. 15-day return/refund guarantee applies. Local sales tax must be included.

Name_____

Address_____
(No shipment to P.O. box addresses without prepayment.)

City_____

State_____ Zip_____

☐ **VISA** ☐ **MasterCard** ☐ **American Express**—Please charge my credit card. Publisher pays shipping/handling. If I return the book(s) at the end of 15 days, charges will be cancelled.

Card #_____ Exp._____

Signature_____

Offer good in U.S.A. and territorial possessions only and subject to credit department approval. Prices subject to change. Prices slightly higher in Canada.

C 8358

VNR VAN NOSTRAND REINHOLD

CIRCLE 44 ON READER SERVICE CARD

The Advanced Programmer's Editor That Doesn't Waste Your Time

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Great on-line help system
- Multiple windows, files
- Regular Expression search
- Unlimited file size, line length
- Supports large displays
- 30 day money-back guarantee
- Not copy protected

Only $195

Lugaru Software Ltd.

5740 Darlington Road
Pittsburgh, PA 15217

Call (412) 421-5911

for IBM PC/XT/AT's or compatibles

CIRCLE 45 ON READER SERVICE CARD

Introductory Offer: $89.50 prepaid, Visa, MC, COD

"Looks like a Query language, to me."

"That, too."

dNPL® / Reporter

A Fourth Generation *Relational* Report Writing System for dBase files with its own full screen editor

dNPL is the *Non-Programmer's Language* for the Business Professional

Tabular and matrix reports, mailing labels, forms, letters
Interactive query for testing data and report options
Simple, English-like, SQL-like sentences

Records selected by multi-function criteria, with values, ranges, strings, wild cards

Up to 10 nested grouping levels with subtotals
Summaries with row and column totals, average, min, max, avg-sum-of-the-squares
Cross-footing with computed totals

Combined sort keys up to 512 characters
Very fast RAM sort, with disk merge for largest files
Matrix reports sorted across page by one variable, down page by others

Up to 255 memory variables
Up to 255 database & defined variables
Conversions of variable types - integer, decimal, money, alpha, date
Trigonometric, date, string, sqr, sqrt, ln, exp functions
Unlimited math expressions & nested If...Then...Else logic
Comparisions between successive records

Link up to 8 databases - dBase II, dBase III, ascii files
Copy, concatenate, type, rename, delete, index, join, rebuild database files

Full screen editor - paging, string search, copy buffer
Create program files for reports & multi-steps reporting procedures
Display menus; prompt user response
Send reports to printer, screen or disk files
Branch, case & repeat statements
User control of formfeed, curser, mutli-color windows, pause
Execute DOS commands, run other programs, from within dNPL

Up to 255 lines per record
Up to 255 data items per line
Multiple reports per page
Unlimited records per report
Unlimited report width

Manuals & primer with hundreds of examples
Not copy protected

Write your first report within minutes

DATABASE APPLICATIONS, INC.

400 Wall Street • Princeton, NJ 08540 • 609-924-2900

Add $3 shipping, $2 COD, 6% NJ tax — 30 day money back guarantee

CIRCLE 46 ON READER SERVICE CARD

tory. Note that a perfectly good primary key may have no meaning whatsoever, possibly even being a randomly generated number (as long as the generator never produced the same number twice).

Operations performed on the data base rarely need to reference the key explicitly. One wants often to retrieve or update tuples where this or that attribute meets some specified test, without reference to the content of a particular key. Nevertheless, it is the existence of the explicit key (and not a link pointer or some other such device) in the relational model that guarantees you can immediately reference each and every tuple in all the relations of a data base.

DBMSs all provide some automatic way of determining that a combination of attributes designated as the primary key are unique as each tuple is inserted into a relation. Not all products provide dictionary or directory-level support for ensuring the uniqueness of a single attribute (except the record number). It is desirable that this support be at the dictionary or directory level.

Many products can enforce uniqueness in single attributes (which may be primary or alternate keys) by forcing the developer to index the key and preventing duplicate entries in the index. This is not a good idea, however, because it creates overhead associated with an otherwise unnecessary index. More importantly, it represents a weak point in the integrity of the DBMS unless the system can ensure that no update can take place at any language level without using the index! Products that support uniqueness integrity at the dictionary or directory level include Revelation, UNIFY, and all mainframe DBMSs.

Another integrity issue has to do with the inherent validity of nonkey attributes: the notion of domains. We have encountered a part of this concept before in the form of the data types supported by DBMS dictionaries. The types *string* and *numeric* are base domains. They are the universe of possibility from which all data is formulated. *Numeric* itself is often broken into two base domains: *INTEGER* and *FLOAT*. Almost all DBMSs support these types. In many products, this is the only level of integrity that cannot be violated. No program can force an update of a tuple when string data is placed in an attribute of numeric type. The reverse also holds true, unless the numeric is being treated as a string (for example, enclosed in quotes).

As types become more sophisticated, the base domain type blurs into the extended domain concept. A good example of this is the *DATE* type supported by many products. Dates can be assimilated in many forms, but their meaning is constant. Note that no notion of domain restriction will ever guarantee that data is

always entered correctly.

Given an American date format, for example, there is no way to determine that the date string "01/02/87" really should have been "02/01/87." Even given a rule which states that the entered date must fall before the current system date, both of these strings would pass for true if the current date happened to be after February 1.

The idea behind domain restrictions is to validate data at the semantic (meaning) level. The relational model is inherently meaning oriented, but base domains are syntactic in nature and embody no meaning in and of themselves (many quantities in the real world can be represented as integers). Some types (like *DATE*) approach a desirable level of power and others (like *MONEY*, a restricted float) are also extensions in that direction. As yet there is no product that supports arbitrary types. Even so, the concept of domain is fundamentally important to the generation of meaningful relations between tables.

The population of Iowa and the number of gallons of milk produced in Iowa in a month may both be expressed as integers. As it stands, only the mainframe-based Supra takes domain names seriously such that it will not allow the production of a join on the logical level between the domains *POPULATION* and *GALLONS*. In Supra, the concept of domain does not go any further than distinguishing between domain names.

Most products are unaware of the semantic potential in names, but they do contain domain integrity support of another kind in the form of edit masks, enumerated types, and range checks. Revelation supports complex edit masking, enumerated types, and range checks at the data dictionary level, while R:base System V supports range checks at the directory level in the form of rules that can be applied to individual attributes of a relation (table). Other kinds of simple edit checks may also be built into the table's directory, though rBase only permits the specification of 20 rules per table.

UNIFY supports extended domain restrictions in its advanced field attributes (AFA). Like the R:base System V rules, these are stored in a system-controlled part of the data base environment and accessed through the system directory. They consist of a list of fields with associated validation rules including range checking, default value assignment, enumerated legal entries and some edit masking. UNIFY also uses the advanced field attributes to generate messages to the operator, both for help with entering valid data and as a response to erroneous data entry. UNIFY has no limit on the

Finally, powerful graphics for Turbo Pascal!

Announcing:
TurboHALO Graphics!

- 150 graphic primitives (based upon the powerful HALO primitives).
- Graphics in up to 16 colors, in medium or high resolution, on 10 popular graphics cards—including **IBM CGA and EGA, Hercules, and AT&T DEB.**
- Output to over 20 printers and plotters—including **HP and Corona Laser and Apple Imagewriter.**
- 11 popular input devices supported.
- Memory resident drivers require only 2K of Turbo Pascal code space.
- These professional graphics programs use **HALO Primitives:**

  Drafix II—Foresight Resources
  CADKEY—Micro Control Systems, Inc.
  Dr. Halo II—Media Cybernetics/IMSI
  CAD Master—Datagraphics
  Decision Master—Centec
  Artworks—West End Films

- Speed, ease of use, multiple fonts and much more, for **ONLY $129.00! NO ROYALTY FEES!**

COMPARE THE FEATURES!

TurboHALO vs. Borland's Turbo Grafix ToolBox:

Feature	TurboHALO	Borland
16-Color Support?	**Yes**	No!
Full EGA Support?	**Yes**	No!
Laser Printer Support?	**Yes**	No!
Multiple Input Drivers?	**Yes**	No!
Do Professional Programs Use Same Primitives?	**Yes**	No!

Call Now To Order!

IMSI™

1299 Fourth Street • San Rafael, California 94901 • (415) 454-7101
Toll Free (800) 222-GRAF (Outside CA) • **(800) 562-GRAF** (In CA)

TURBOHALO is published by IMSI and is a registered trademark of MEDIA CYBERNETICS, INC.

CIRCLE 47 ON READER SERVICE CARD

number of AFAs that may apply to a given table.

Other products that support masks, enumerations, and/or ranges do so through their screen forms, *PICTURE* statements in their *ACCEPT* or *GET* verbs, and/or triggered procedures executed by entry programs.

Triggered procedures are programs or subprograms executed before or after data update. Such programs may do elaborate entity and referential data checking. In theory, triggered procedures could accomplish any and all kinds of data validation, but because overhead is associated with their use, it is better to use higher-level constructs like dictionary-based enumerated types or screen-based constructs, when available.

All DBMS products provide for triggered procedures if one is coding at a low-enough level (they are often simple subroutines). Revelation (the only product that can trigger procedures directly from the dictionary), Data Access Corp.'s DataFlex, and Informix-4GL all contain direct support for triggered procedures from high-level entry screens, but R:base System V, Paradox, dBASE III+, and UNIFY do not.

Ideally, the domain concept should be free of edit-like restrictions and represent meaning only. The domain *distance* could be miles in one attribute and kilometers in another. An attempt to join relations on this attribute should automatically invoke conversion routines at the dictionary or directory level that would make the join meaningful. To date, no product has been able to achieve this. The problem of meaning is difficult to compute. I suspect it will be achieved to a greater extent in the future, but not without consuming large amounts of CPU power—like graphic interfaces!

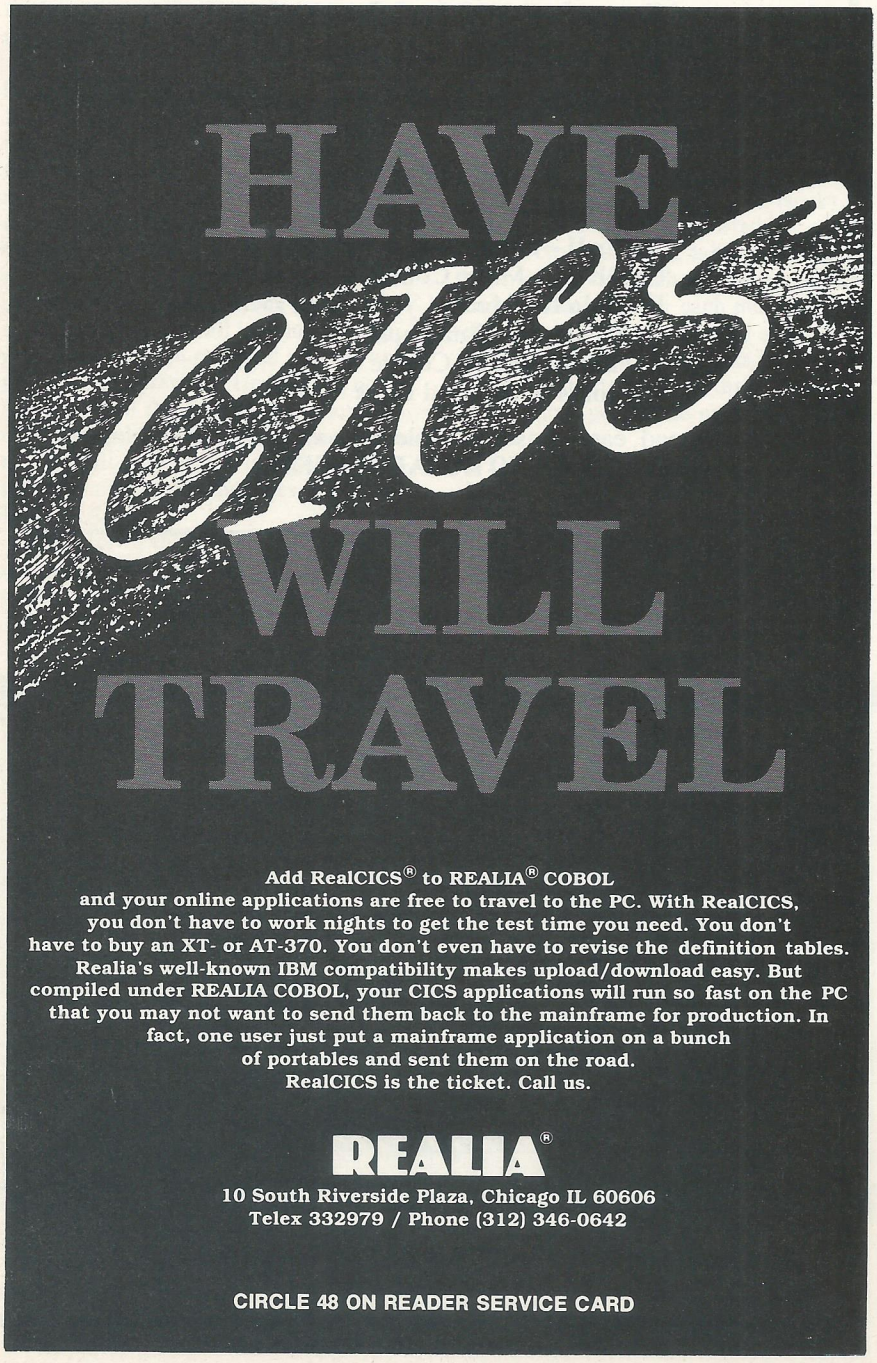
* Like the integrity rule, there is also a referential rule. This rule states that no tuple may contain a value in a foreign key if that value is not already present as a primary key in some other relation. A foreign key is a link between the tuples of one table and the primary key of another. It is the relational mechanism for producing meaningful joins between tables. It should be noted that the foreign key and the primary key to which it refers must have the same domain if the join is to be meaningful.

An example of the two keys having the same domain is that of a tuple in an *EMPLOYEE* relation that contains an attribute called *JOB.NUMBER*. *JOB.NUMBER* is a foreign key in *EMPLOYEE* because it happens to be the primary key in the relation *JOBS*, which has as attributes minimum and maximum salary, job descriptions, and other things.

If we know that *EMPLOYEE* Mark Jones has *JOB.NUMBER 107*, we are guaranteed that *107* will return one tuple only from the *JOBS* relation. The foreign key cannot return more than one tuple in *JOBS* because of the uniqueness rule of primary keys. It will not return less than one tuple because the referential rule prevents us from assigning *107* as any person's job number unless that number exists already as a primary key in *JOBS*. Therefore, because of the two rules we can always find the job description (among other things) for job 107 without having to store the description directly in the *EMPLOYEE* relation. Figure 2 illustrates some of the possible relationships between foreign and primary keys.

Besides the referential rule, referential constraints may exist between an attribute and other attributes in the same tuple (record constraints), or among attributes in different tuples of the relation (set constraints).

Each of these constraints can be further broken down into state, transition, and deferred constraints. In a state constraint, the immediate validity of the attribute is of concern; for example, "'*A*' must be greater than 1." Transition constraints have to do with the effect of the attribute on the tuple or relation after it



Now COBOL Programmers Can Do Formatted Screens Quickly and Easily - with SCREENIO.

SCREENIO is the Screen Manager for Realia™ COBOL.

SCREENIO is a high-performance, COBOL-specific screen manager for the Personal Computer and compatibles. It's powerful, offers an impressive array of features, yet is easy to use. SCREENIO was written by experienced COBOL professionals and represents a truly COBOL approach to screen management on the PC.

Design and Revise Screens Painlessly With Our Panel Editor.

Use our interactive Panel Editor Facility to design your screens with a What-You-See-Is-What-You-Get approach. You can easily modify and experiment with screen layouts. Type titles and descriptive information on the screen, draw lines and boxes using the cursor movement keys, and paint the colors you want using the cursor keys or block functions. Use all 256 color combinations on your screens.

Define your data fields in COBOL terms; use your own data name, PICTURE and VALUE clauses. Specify the field type (various alpha and numeric types, hex, etc.), valid entries for the field (if you want SCREENIO to edit your data for you), and various other items. The Panel Editor will produce a copybook that contains this information. Panel revisions rarely mean changing your source code.

Programming for SCREENIO is Easy.

You can display your screen and accept data with as few as two statements; a COPY *panelname* in your WORKING-STORAGE SECTION, and a CALL SCREENIO statement in the PROCEDURE DIVISION. SCREENIO is linked with your application just as any other COBOL subroutine would be.

SCREENIO takes care of tabbing, data validation, and other functions expected of a good screen manager. When we return to your COBOL program, the data from the user fields on the screen is in your variables.

Yes, we've always done windows. SCREENIO supports Edit Masks, Automatic Error Detection and Editing, High Speed Video, Hot-Fields, Key Redefinition, Foreign Language Capability, Custom Tabbing Order, Color Control at Runtime, PC Speaker Control, and much, much more. A BMS Mapset Import is available. Most programmers are amazed at how easy it is to build flashy applications using SCREENIO and COBOL.

Our Support is Outstanding.

We provide superb telephone support. Upgrades are distributed to licensed users automatically for the first year. There are no Runtime Fees - the code you develop is yours to distribute freely.

Call us to discuss your application. Get a feel for the kind of quality support we provide.

We'll Send You a Free Demonstration Diskette.

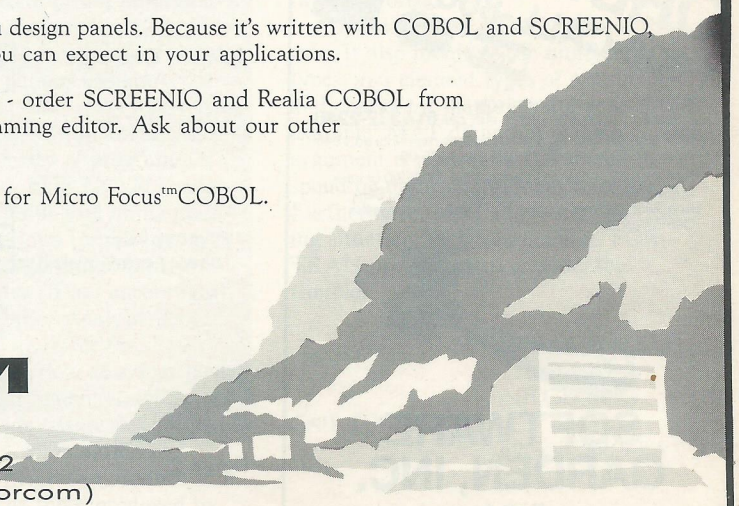
This limited version of our Panel Editor shows how you design panels. Because it's written with COBOL and SCREENIO, it also shows the kind of features and performance you can expect in your applications.

SCREENIO 2.1 is only $400 plus shipping. Save $100 - order SCREENIO and Realia COBOL from us. We also offer packages with Kedit, a great programming editor. Ask about our other programming tools, too.

Not satisfied with FORMS™? Ask about SCREENIO for Micro Focus™ COBOL.

In a hurry? Most orders are *delivered* within 24 hours!

- Amex MC Visa Check -



NORCOM

Northern Computing Consultants
P.O. Box 020897 • Juneau, Alaska 99802
(907) 780-6464 • TELEX 510 601 4951 (Norcom)

CIRCLE 49 ON READER SERVICE CARD

is updated (for example, "'*A*' must increase"). Deferred constraints may be violated for some period of time but must be satisfied after some prespecified series of transactions is completed. Two or more values may be allowed out of balance until after some predefined set of transactions has been completed.

As in other things, the level of support for referential integrity varies widely among DBMS products. UNIFY, R:base System V, and the mainframe products have some support at the directory level. Only UNIFY (to my knowledge) supports the integrity rule directly in its data dictionary. Most of the other products provide what support they have through their entry screens and via triggered procedures.

When applications surrounding a data base have a stable set of input channels, externally applied integrity rules are usually sufficient. But when the data base environment is sufficiently large, encompassing a number of departments, each department develops new input requirements that may impact similar (but unaccounted for) developments in other departments.

In large corporations where DBMSs may be used to automate business spanning many departments, with dozens or even hundreds of application developers, one cannot guarantee that the application of external rules will always be followed rigorously or that the rules that are followed are sufficient to meet the needs of departments whose requirements may evolve along different lines.

Very large-scale development is impractical without at least some support for integrity rules in the dictionary, but department-wide (or small-company-wide) data base automation has been successfully implemented without it.

References

Date, C. J. *An Introduction To Database Systems*, Vols. I and II. Addison-Wesley Pub., 1986.

Maier, David. *The Theory of Relational Databases*. Computer Science Press, 1983.

Martin, Daniel. *Advanced Database Techniques*. The MIT Press, 1986.

Papadimitriou, Christos. *The Theory of Database Concurrency Control*. Computer Science Press, 1986.

Wertz, Charles. *The Data Dictionary, Concepts and Uses*. QED Information Sciences, 1986.

Possible relationships between foreign and primary keys

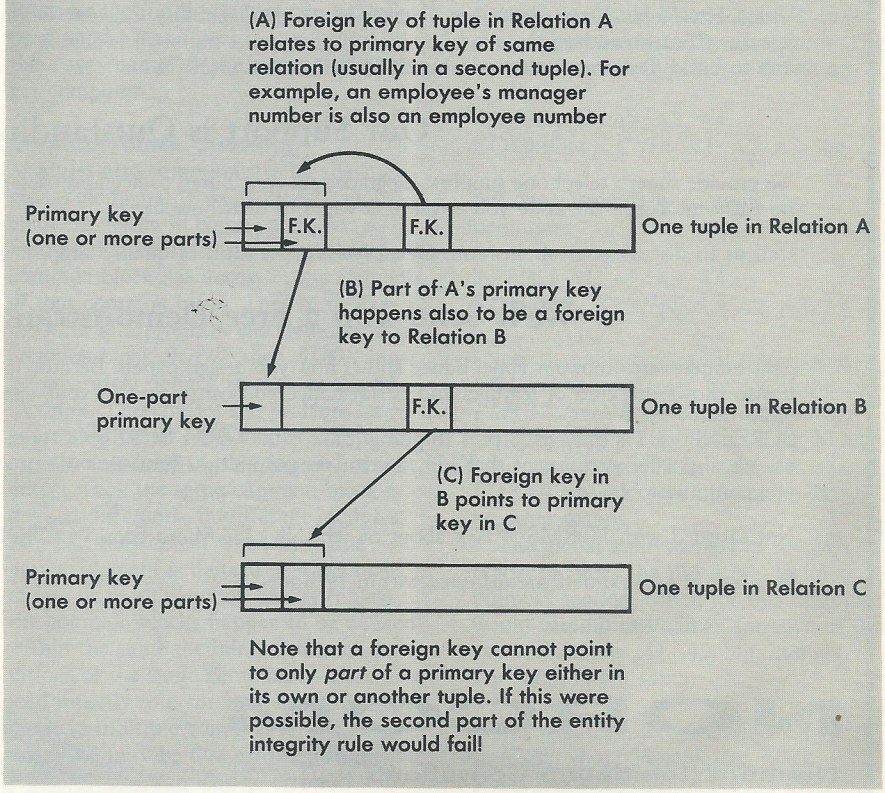


Figure 2.

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

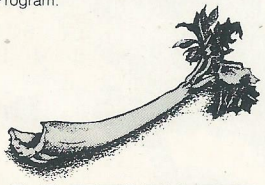
Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! JUST $49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!
1-800-CALL-800 x8088
VISA MasterCard AMERICAN EXPRESS

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**
No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.
Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.

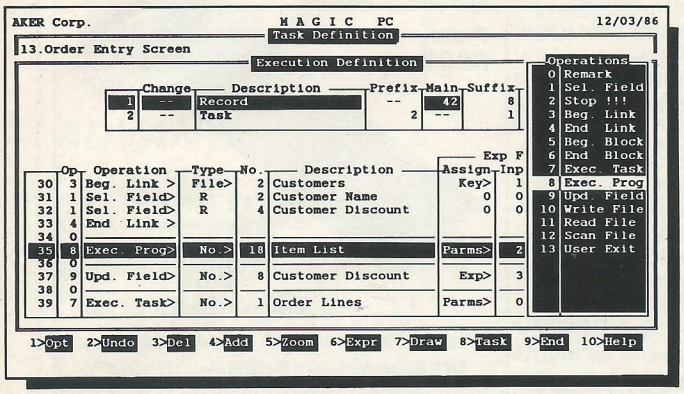


SOFTWARE GARDEN, INC.
Dept. L
P.O. Box 373, Newton Highlands, MA 02161

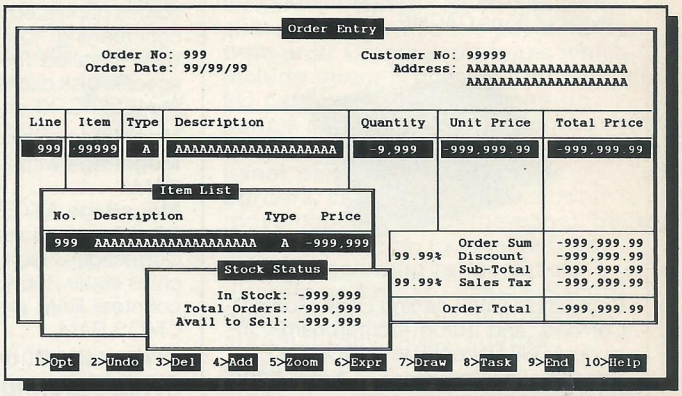
CIRCLE 50 ON READER SERVICE CARD

Announcing Magic PC – the first breakthrough for database applications developers in over 20 years:

Now you can develop professional applications 1000% faster than your 4GL or DBMS, totally free from programming, commands and syntax!



A Magic PC program looks as simple as this. To design an application you quickly fill-in menu-driven decision tables **without having to write a single line of code.** For example, just by highlighting the Execute Program operation on this screen and also highlighting the Item List program in the Program Menu, you tell Magic PC to pop-up the Item List window shown in the adjacent screen, when the end-user hits the Zoom key.



Magic PC gives your end-user the power to harness and retrieve data instantly, **without any commands or syntax** because at runtime you already have built-in options to Add, Delete, Modify, Query and get on-the-spot ad-hoc information simply by highlighting selections from menus. Dats validation, security and error-checking are done automatically for you by Magic PC without programming.

Who needs another DBMS?

At last, Magic PC gives you the ultimate applications design tool, far ahead of 4GL's, DBMS and Application Generators.

Magic PC breaks through the language barrier with the revolutionary Un-Language concept:

NO PROGRAMMING, COMMANDS OR SYNTAX!

Free yourself from your programming language

Magic PC makes you, the professional, completely free from the drudgery of procedural programming. No more cryptic commands, syntax or unforgiving procedural structures, because Magic PC does all the programming automatically. There's your competitive edge. The rest is up to you . . .

The Professional Choice

Already an international success, Magic PC is a profit maker and career booster for DP Consultants, System Integrators, VARs, MIS professionals, System Analysts, Programmer Analysts and Software Engineers. If you design PC applications professionally, you can't afford not to Un-Language now.

IBM France: "IBM encourages this introduction and can not help but salute such evolution . . ."

Israeli Air Force: "We were convinced that it was not possible to have a design tool powerful enough to implement real-life applications without a programming language. Magic PC changed our mind . . ."

Jeff Duntemann, PC Tech Journal: "It's probably the best integrated database applications and screen generator that I have ever seen . . .very smooth system, and smoothness comes at a premium these days . . ."

The Magic PC Secret

You're so much more productive with Magic PC because there is **absolutely no programming** to slow you down. You design a Magic PC application by simply filling-in the **Data Dictionary Tables** (Files, Fields, Keys) and the **Task Description Tables** (Operations and Expressions).

Only 13 design **Operations** harness the power of Magic PC. Operations are specific enough to eliminate the need for tiresome syntax, yet elastic enough to produce robust custom applications. Use the Operations to describe **what** you want and Magic PC makes it happen. It's that simple.

Make Task nesting power available with a single **Execute Task** Operation. This powerful instruction triggers Magic PC to execute and display additional tasks or even external applications through **Window Zooms.** The 3-dimensional effect of Window Zooming lets you probe deep into your application through nested windows and manipulate the data underneath.

You describe a Magic PC Task or Program (composite Tasks) by filling your system analysis flow into the Task Description Tables. Choose the participating Data View, and Magic PC executes your desired Operations. You interface with the Tables by highlighting your selections from pop-up menu-driven windows. There's nothing to edit except your headings.

You're not confined to any particular design sequence as you are with most procedural languages. You can enter and change any Table spontaneously, on the fly, as ideas come to mind and Magic PC automatically maintains the application integrity.

A **Magic Inference Engine** automatically orchestrates your Task Description Tables into a single file of internal **Knowledge Base Rules** for optimum, bug-free performance. Knowledge Base Rules are executed by the **Magic Run** engine for stand-alone runtime operation, or by the **Magic Lan** engine for unrestricted Novell network sharing. You're free to design the Knowledge Base without worrying about the internal structure.

Discover fast, language-free programming at no risk for only $19.95

See for yourself how fast you can program language-free applications with our low-cost limited offer.

You'll get the full Magic PC software unprotected and limited to 100 records and 450 page documentation complete with a **free** Order Entry sample application. You'll also get our **free** telephone support for 90 days!

And your $19.95 will be credited towards the full $695 Magic PC purchase price. Even if you don't buy Magic PC right away, keep your $19.95 Magic PC Trial as your application prototyping tool at this bargain price.

Our No-Risk Guarantee!

You have our no-risk 30-day money-back guarantee: if you're not completely satisfied for any reason, even Magic PC Trial for $19.95, send it back for a refund.

Order now while supply lasts

Call this toll free number now with your Visa, MasterCard or American Express for immediate delivery, or send the Order Coupon below today to Aker.

1-800-345-MAGIC
in CA call 714-250-1718

VISA MasterCard AMERICAN EXPRESS

Yes, please rush me:

☐ Magic PC Trial	$ 19.95
☐ Magic PC	$695.00
Add shipping	$ 5.00
In CA 6% tax	$_____
Prices valid in US only.	Total $_____

Ship to:_____

Address:_____

City/ST/Zip:_____

Phone:_____

AKER

Aker Corp. 18007 Skypark Circle B2, Irvine, CA 92714
(714) 250-1718, Elec. Mail Dialcom 41:AKR 001 Telex 4931184
AKR UI OEM and VAR inquiries are welcome.

Min. requirements PC DOS 2.0, IBM PC or 100% compatible with 512K and hard disk.

©1986 Aker Corp. Printed 1/87 **Trademarks:** Magic PC, Un-Language, Window Zoom, Magic Run, Magic LAN and Magic PC Trial are trademarks of Aker Corp., IBM PC and PC-DOS are trademarks of IBM Corp., Novell is a trademark of Novell Inc.

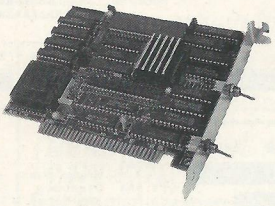
CIRCLE 40 ON READER SERVICE CARD

MICROWAY ACCELERATES YOUR PC!

FastCACHE-286™

Runs your PC Faster than an AT!

Runs the 80286 at 9 or 12 MHz and the 80287 at 8, 9 or 12 MHz. Includes 8 kbytes of 55ns CACHE.



Compatible with Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache, Print Spooler and Diagnostics... **From $399**

LOTUS/INTEL EMS SPECIFICATION BOARDS

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles...**$549**

MegaPage with ϕK..... **$149**

MegaPage with 2 megabytes of HMOS RAM..... **$419**

MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. With 1 megabyte CMOS RAM..... **$699**

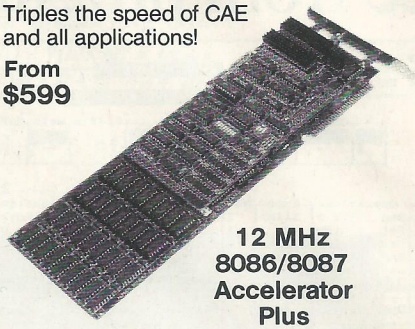
INTEL, JRAM, or Maynard ..... **CALL**

INTEL INBOARD 386 ϕK..... **$1325**

NUMBER SMASHER/ECM™

Triples the speed of CAE and all applications!

From $599



12 MHz 8086/8087 Accelerator Plus A Megabyte for DOS!

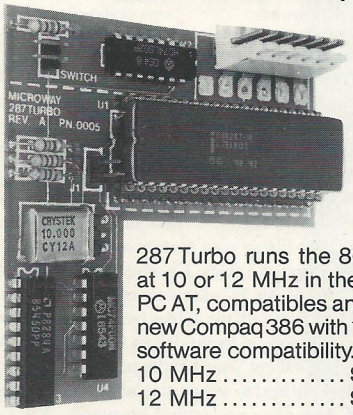
For the IBM PC, XT and compatibles

***PC Magazine* "Editor's Choice"**

8087 SOFTWARE

IBM BASIC COMPILER	$465
MICROSOFT QUICK BASIC	$79
87BASIC COMPILER PATCH	$150
87BASIC/INLINE	$200
IBM MACRO ASSEMBLER	$155
MS MACRO ASSEMBLER	$99
87MACRO/DEBUG	$199
MICROSOFT FORTRAN V4	$299
RM FORTRAN	$399
LAHEY FORTRAN F77L	$477
MS or LATTICE C	CALL
STSC APL★PLUS/PC	$450
STSC STATGRAPHICS	$675
SPSS/PC+	$695
87SFL Scientific Functions	$250
87FFT	$200
OBJ → ASM	$200
PHOENIX PRODUCTS	CALL

287 Turbo™-10/12



287 Turbo runs the 80287 at 10 or 12 MHz in the IBM PC AT, compatibles and the new Compaq 386 with 100% software compatibility.

10 MHz **$450**

12 MHz **$550**

***PC Magazine* "Editor's Choice"**

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087 5 MHz..... **$105**
For the IBM PC, XT and compatibles

8087-2 8 MHz..... **$154**
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5 MHz..... **$179**
For the IBM PC AT and 286 compatibles

80287-6 6 MHz..... **$229**
For 8 MHz AT and compatibles

80287-8 8 MHz..... **$259**
For the 8 MHz 80286 accelerator cards and Compaq 386

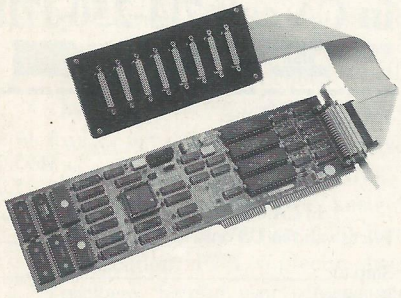
80287-10 10 MHz..... **$395**

PC-PAL™ Programmer..... **$395**

Call for great prices on V20, V30, 64K, 128K and 256K RAM

AT8™

Turns your AT into a high speed, multi-user Xenix business system!



8 port, intelligent serial controller with 3% response degradation. Includes 8 MHz 80186 with built in DMA **$1299**

MICROWAY SOFTWARE FOR LOTUS 1-2-3™

FASTBREAK™ employs the 8087 to increase the speed of Lotus 1-2-3™ Version 1A or 1A*. Users are reporting speed ups of between 3 and 36 to 1. When run with our NUMBER SMASHER accelerator card, recalculation speed ups of 10 to 30 are being reported..... **$79**

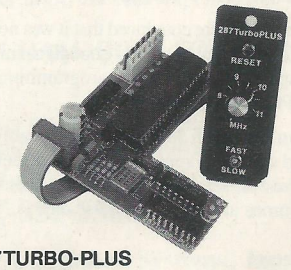
PowerDialer® Add-In for Lotus 1-2-3 Release 2. Automated telephone dialing from within 1-2-3. Adds least cost routing, automatic carrier selection and automated phone book worksheet. Builds customized dialing applications. Can be used with DesqView **$79**

HOTLINK™ adds easy linking of spreadsheets to Lotus 1-2-3 Version 1A... **$99**

287 TURBO-PLUS™

Speeds up your AT

Adjustable 80286 Clock 6-12 MHz
10 MHz 80287 Clock
Plus Full Hardware Reset..... **$149**
Optional 80286-10 ..... **$175**



287TURBO-PLUS
With 80287 10 MHz..... **$549**
With 80287 12 MHz..... **$629**

CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

MicroWay
P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

You Can Talk To Us!

MicroWay Europe
32 High Street
Kingston-Upon-Thames
Surrey England KT1 1HL
Telephone: 01-541-5466

CIRCLE 52 ON READER SERVICE CARD

EXOTIC LANGUAGE OF THE MONTH CLUB

NPL: A nonprogrammer's data base language

By Thomas D. Truitt

If you are thinking about building a data base system to coordinate data from several sources, you have already begun to think in higher-level terms.

As a programmer, I get lost too often in a thicket of bits, bytes, and records, unable to see the forest for the trees. As a nonprogrammer—which is what I am most of the time, being too busy to indulge in the pleasure of creating elegant algorithms for esoteric programming problems—I need programming tools to build fairly elaborate record-keeping and interactive business systems. And I need to be able to do prototype testing in a few hours or a couple of days. With my global thinking hat on I want to visualize grand solutions to big problems and then single-handedly test and demonstrate the theory with real data before the client loses interest.

The needs of nonprogrammers are not the same as those of novices. Nonprogrammers are busy people with exacting requirements, or they are professional programmers and system designers using higher-level languages for greater personal productivity. The marriage of data base management software using nonprocedural methods with Ramis (1969), SQL (1974), Focus (1975), and other fourth generation language data base systems provided a new kind of operating environment, where early results have greater importance than minimum consumption of computer resources. In the preface to his book, *Applications Development Without Programmers*,[1] James Martin writes, "Applications development did not change much for twenty years, but now a new wave is crashing in. A rich diversity of nonprocedural languages are emerging."(sic)

We will look at the criteria for a nonprogrammer's data base language, but first a few comments on the general objectives that guided the evolution of the NPL system. NPL's roots are found in the technology of the mainframe fourth generation languages mentioned earlier. The use of three principal phrases such as, *print x y z*, *if a > b*, and *sort by m*, is central to the query processing and report writing of each of those early systems. In NPL, the *print*, *if* and *by* phrases of report requests form the basis for most of the processing of data files.

NPL has grown from its first implementation in 1980 to become a full-scale relational data base applications system. The basic design plan was published in book form$_2$ in 1983. The following considerations have influenced every design decision along the way.

Programming languages vs. query language. A query language is used to extract data interactively. The requirements for such a language are the same as for a nonprogrammer's language. Program statements should be as similar as possible, whether typed at the console or stored in a program file. A program file should be able to duplicate everything that can be entered at the keyboard, except perhaps the cursor control keys and Control-break. The reverse need not be true, for the interactive user would not expect to duplicate certain program statements, such as those that loop on a group of stored statements and prompt the user for input.

Let the user make program changes readily. This is a subjective judgement, but if the user is developing a data base application, he or she must be able to experiment with alternative files, field elements, formats, and various program options. Open access to all of these elements is better than allowing the system too much control over the elements of the application.

In NPL, open access means that each data base file is simply a DOS ASCII file, the formal description of each data base file is stored in another ASCII text file (one record per data field), and every program is a separate text file. Other text files are used to save groups of program variables. The developer may examine and edit all files except the index files, and the end user may be denied such access.

Program readability. The keywords of a language must not be abbreviated beyond recognition by a beginner. NPL keywords may not be abbreviated, with a few familiar exceptions. Since few programmers take the time to explain their programs, statements should read more or less like English text to minimize the need for explanatory comments.

NPL offers a completely free format for write commands and NPL sentences. Multiple commands can appear on a line. No continuation codes are needed. Commas and semicolons may be inserted freely to improve readability. Also, the syntax is permissive, not rigid; phrases in an NPL sentence may appear in any order, and many keywords have one or more synonyms or alternative spellings.

Enough tools for the experienced. There is a challenge in balancing simplicity and conciseness with power and flexibility. Ideally, a nonprogrammer's language has all of these traits without compromising one extreme for the other. Experienced programmers use higher-level instructions (commands that take giant strides and perform many routine functions with only a few words), provided they can override the standard functions with additional words to achieve special effects. Thus there must be a few commands that will satisfy nonprogrammers' needs, plus a rich list of variants to the standard options—and all of this must make some sense to the person who can only read English.

For example, this sentence prints a simple listing of data arranged in columns:

```
PRINT NAME EMPID# DEPT SEX
   AGE SALARY.
```

The next sentence produces a two-way table of summary statistics from the same data file:

```
WRITE AVE.AGE MAX.SALARY
   AVE.SALARY MIN.SALARY,
   ACROSS SEX, BY DEPT.
```

And this one prints mailing labels:

```
PRINT NAME OVER COMPANY,
   OVER STREET, OVER CITY ST
   AND ZIP, SKIP 5.
```

Structured programming. NPL uses structured programming. The language adopts the position that programmers and nonprogrammers should write procedural statements in a way that flows down the page and jumps back up only as part of a *REPEAT...UNTIL* statement. In other words, the dangerous *GOTO* statement, used for jumping around in programs, is not permitted in NPL. This restriction encourages structured programs, which are much easier to understand after the nonprogrammer has forgotten why a procedure was written in a certain way.

Now let's look at the important elements of NPL. But before reviewing the various statements available to the user, consider a few broad concepts. Look at Figure 1, the NPL/R System Guide diagram, as a floor plan of several rooms in which one may use only the commands listed for each particular room. A period is used to leave a room and return to the command mode room, in which all statements are imperative commands of one, two, or three words.

The operational modes are described briefly as follows:

- Command mode, for miscellaneous controls and file operations
- Report-writing mode, also for writing new data files
- Define mode, for defining new table variables
- Set mode, for defining new memory variables
- Select mode, to specify the current index and a value range or list
- Update mode, for batch processing changes to data files
- Three modes for interactive editing of files: *Modify*, for editing data files; *Edit*, for editing a program and master files; and *Create*, for creating and editing master files.

The highest-level command in the NPL language is *DO <filename>*, which calls a program file for execution. In the MS-DOS versions of the product, this file may be an NPL program file or a DOS .COM, .EXE, or .BAT file. Any NPL program file may contain one or more *DO* commands, and these calls may be nested to any depth. *CALL:* is a similar command to call a subroutine within a program file.

Flow within a program file is controlled by four procedural statements:

1. Loop control:

```
REPEAT: <statements>
        <statements>
                ...
UNTIL: N EQ: 5
```

2. Case selection:

```
FOR: N
     EQ: 1 <statements>
     EQ: 2 <statements>
     LE: 5 <statements>
     ELSE: <statements>
RESUME:
```

where *N* is some program variable.

3. The *QUIT:* command, which transfers control to the program file named by *QUITEXEC*, a *SET* variable. Alternatively, *EXITBLOCK:* jumps to the next *RESUME:* or *UNTIL:* command at the end of a *FOR:* or *REPEAT:* block. If *EXITBLOCK:* is followed by an integer *n*, the program exits from *n* nesting levels of *FOR:*/*REPEAT:* statements.

4. The *WAIT:* command, which stops the program and waits for the user to read the screen and press a key to proceed.

These few keywords and *DO* are all the nonprogrammer needs to remember about program flow.

It is useful to think of three kinds of statements in NPL: nonprocedural, procedural, and assignment.

Nonprocedural statements contain the user's requirements—that is, what the

Now with extended/expanded memory support

PC Scheme—
a simple, modern LISP
for under $100.

Texas Instruments presents PC Scheme, the $95* solution to your symbolic processing needs.

Whether you're an experienced LISP user, or just beginning to discover the power of symbolic programming languages, PC Scheme is the right product for you. It runs on IBM® Personal Computers, as well as the TI Professional Computer family, including the Business-Pro™ computer.

Powerful features include an optimizing incremental compiler for ease of programming and fast execution; an EMACS-like editor; extensions for debugging, graphics, and windowing; DOS-CALL capability; and a programming system for the development of object-oriented applications—all designed to work efficiently on personal computers.

To order, or for more information, call toll-free:

1-800-527-3500

*Suggested list price.
Business-Pro is a trademark of Texas Instruments Incorporated.
IBM is a registered trademark of International Business Machines Corporation.

TEXAS INSTRUMENTS

Creating useful products and services for you.

© 1986 TI 261765-02A

NPL/R Plus system guide

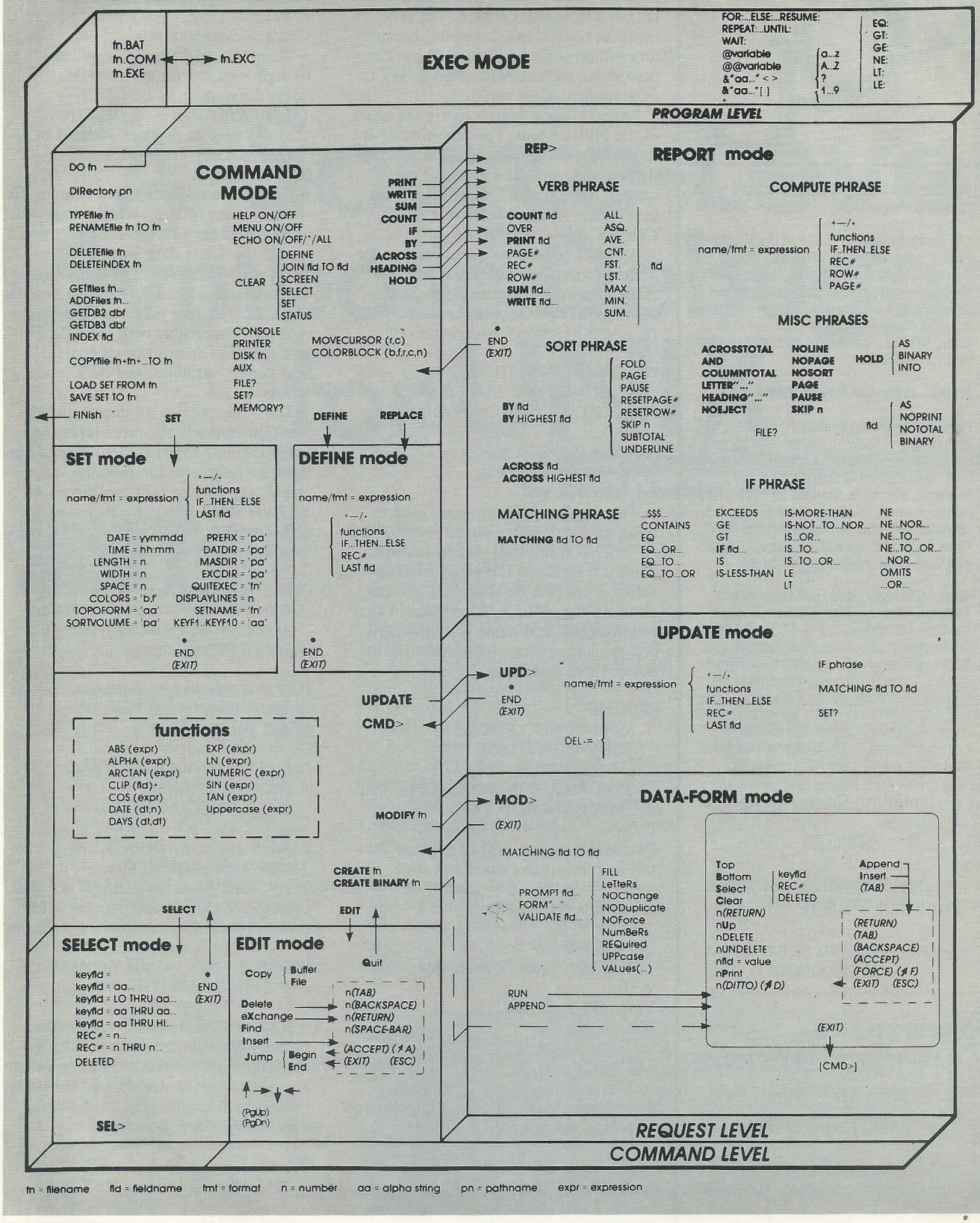


Figure 1.

CQL

SQL Compatible Query System adaptable to any operating environment.

CQL Query System. A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

Including Source Code

$395.00

File System interfaces include C-tree and BTRIEVE.

HARDWARE AND FILE SYSTEM INDEPENDENT

KURTZBERG COMPUTER SYSTEMS

41-19 BELL BLVD.
BAYSIDE, N.Y. 11361

VISA/Master Charge accepted
(718) 229-4540

*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp.
MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL Logo are trademarks of Kurtzberg Computer Systems.

CIRCLE 53 ON READER SERVICE CARD

output of a process should look like: either the specification of a report or new data file or the graphic and logical details of a data entry screen. The statements are called nonprocedural because no mention is made of the step-by-step procedures the computer must follow to achieve the desired result.

Report request sentences are the heart of the NPL system. The report mode is rich with sorting, screening, and formatting options, and yet a sentence need contain only one *PRINT* phrase. A single primary verb is required: *PRINT, COUNT, SUM,* or *WRITE.*

A report request produces either a formatted report or a new file (using the *HOLD* phrase) for subsequent processing. Typical sentences contain a *PRINT, IF,* and *BY* phrase for sorting, as in:

```
PRINT NAME AND TELEPHONE,
   BY COMPANY, IF STATE IS CA.
PRINT ORDER# CUSTOMER QTY PRICE
   DATE,
   IF TERMS NOT CASH,
   HOLD AS INVOICES.
```

The simplest NPL program used to bring up a data entry screen is *MODIFY APPEND.* Additional statements will paint the screen as the user wishes and validate the data items as they are entered. In the *VALIDATE* statement that follows, each field name is followed by one or more options that control the input of those data items:

```
VALIDATE
   SSN NBR NODUPLICATE,
   AGE VALUES(21..65),
   CITY REQUIRED,
   STATE REQUIRED UPPERCASE LETTERS,
   ZIPCODE NUMBERS REQ FILL, ...
```

Procedural statements include flow control and imperative command statements. Only these flow control statements are needed:

```
DO <filename>
   or CALL: <subroutinename>:
WAIT:
FOR:...ELSE:...RESUME:
REPEAT:...UNTIL:
QUIT:
EXITBLOCK:
```

Three examples of the many imperative command options are:

```
COPYFILE SALESJAN + SALESFEB
   + SALESMAR TO SALESQ1
CLEAR SCREEN
DELETEFILE OLDSTUFF.DAT
JOIN EMP.ID# TO SALARY.EMPID#
```

Assignment statements are neither nonprocedural nor procedural; they simply assign values or expressions for computing values to names of variables. The statements have the general form: *<variable name> / <format> = <value or expression>.*

When the */<format>* is omitted, a 10-digit format with two fractional places is assumed, as in: *COST = (QTY*PRICE)*(1 − %DISC/100).*

Modes vary as they are affected by assignments. Only four modes allow assignment statements:

DEFINE mode assignment statements are definitions of new variables that are evaluated as each record is read during a pass of the active data base, in report mode or update mode. The definitions remain active until a different data base is activated. The new variable names are used interchangeably with the data file field names. The expressions contain field names, variables, and *SET* mode variables.

The latter point is an important feature that connects the procedural and nonprocedural domains. That is, in a dialogue with the user the procedural statements may save certain parameters as *SET* variables, which are then used to define some of the nonprocedural variables. For example:

```
DEFINE FULLNAME/A30 =
   CLIP(FNAME)+' '+MI+' '+LNAME,
   AGE/I2 = SET.YEAR-BIRTHYR.
(Year is a SET variable)
```

DEFINE mode expressions may include variables from several files that are joined. For example:

```
DEFINE COST = (SALES.QTY *
   PRODUCT.PRICE)*(1 −
   TERMS.DISCOUNT/100).
```

where *SALES, PRODUCT,* and *TERMS* are names of files.

UPDATE mode assignments are executed immediately and only once, for all records selected in the specified data file:

```
UPDATE SALARY = 1.08*SALARY,
   IF DEPTCODE IS ENGNRG.
```

SET mode assignment variables, or memory variables, are associated with the procedural domain. The assignments are executed immediately, and the values remain until changed by another *SET* mode assignment statement:

```
SET A=7.9, B=34.3, C=55,
   X=(−B+SQRT(B*B−4*A*C))/2*A.
```

SELECT mode assignment statements apply only to indexed variables. These statements are different in that they do

not allow formats or expressions, only a value or list of values. The values are stored and used during each report or update mode pass of the active data base to select only those records with listed values of the active index:

```
SELECT ZIP = 03000 THRU 06999,
   10000 THRU 19999.
```

Now we can examine the NPL statements according to the object of the action taken by the statements.

The four kinds of objects considered are *data tables* (rows and columns of data items stored in data files, taken as a group and not as individual items), *files* of any type as defined by DOS, *program variables* as specified in the *SET* mode, and the *console cursor* and *screen attributes* (such as color and inverse-video).

Certain statements use data tables as objects. In doing so they automatically read the active data base files, processing all or selected records without further ado.

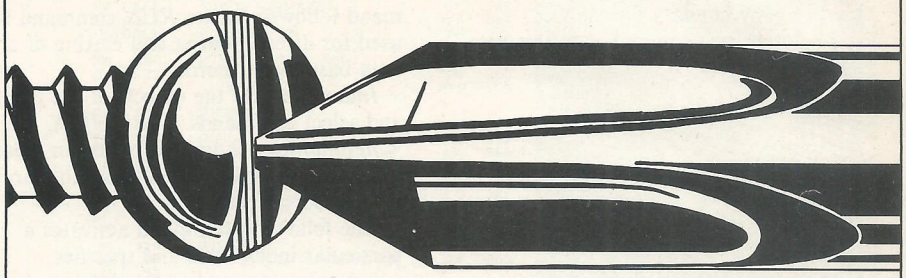
Report request sentences designed to extract data and print reports are:

```
- PRINT <verb object list>
   <IF phrases> <other phrases>.
- WRITE <verb object list>
   <IF phrases> <other phrases>.
- COUNT <verb object>
   <IF phrases> <other phrases>.

DEFINE <variablename>/<format>
   = <expression>.
REBUILD <filename>
UPDATE <fieldname>
   =<expression>, <fieldname>
   =..., <IF phrase>.
            change the data
INSERT <fieldname>
   =<expression>, <fieldname>
   =..., <IF phrase>.
            insert new records
DELETE <IF phrase>.
            delete records
MATCHING <fil1>.<keyfld> TO
   <fil2>.<keyfld>
            join temporarily
JOIN <fil2>.<keyfld> TO
   <fil1>.<keyfld>
            join permanently
```

Seven commands use files as objects. In a few cases the NPL keyword may be abbreviated to the uppercase part of the word, as in these examples:

```
DIRectory <volume or DOS filespec>
TYPEfile <filname>
RENAMEfile <filname> TO <filname>
DELETEfile <filname or DOS filespec>
COPYfile <filname> + <filname>
   +... TO <filname>
```


ISN'T IT A PITY...



Everything Isn't As Accommodating As

c-tree™ / **r-tree**™

FILE HANDLER — REPORT GENERATOR

Performance and Portability

For all the time you devote to developing your new programs, doesn't it make sense to insure they perform like lightning and can be ported with ease?

c-tree: Multi-Key ISAM Functions For Single User, Network, & Multi Tasking Systems

Based on the most advanced B+ Tree routines available today, **c-tree** gives you unmatched keyed file accessing performance and complete C Source Code. Thousands of professional C programmers are already enjoying **c-tree**'s royalty-free benefits, outstanding performance, and unparalleled portability.

Only FairCom provides single and multi-user capabilities in one source code package, including locking routines for Unix, Xenix, and DOS 3.1., for one low price! In addition, **c-tree** supports fixed and variable record length data files; fixed and variable length key values with key compression; multiple indices in a single index file; and automatic sharing of file descriptors.

r-tree: Multi-File Report Generator

r-tree builds on the power of **c-tree** to provide sophisticated, multi-line reports. Information spanning multiple files may be used for display purposes or to direct record selection. You can develop new reports or change existing reports without programming or recompiling and can use any text editor to create or modify **r-tree** report scripts including the complete report layout. At your option, end users may even modify the report scripts you provide.

Unlimited Virtual Fields; Automatic File Traversal

r-tree report scripts can define any number of virtual fields based on complex computational expressions involving application defined data objects and other virtual fields. In addition, **r-tree** automatically computes values based on the MAX, MIN, SUM, FRQ, or AVG of values spread over multiple records. **r-tree** even lets you nest these computational functions, causing files from different logical levels to be automatically traversed.

Unlike other report generators, **r-tree** allows you to distribute executable code capable of producing new reports or changing existing reports without royalty payments, provided the code is tied to an application. Your complete source code also includes the report script interpreter and compiler.

How To Order

Put FairCom leadership in programmers utilities to work for you. Order **c-tree** today for $395 or **r-tree** for $295. (When ordered together, **r-tree** is only $255). For VISA, MasterCard and C.O.D. orders, call 314/445-6833. For **c-tree** benchmark comparisons, write FairCom, 2606 Johnson Drive, Columbia, MO 65203.

Complete C Source Code & No Royalties!

Xenix is a registered trademark of Microsoft Corp. Unix is a registered trademark of AT&T.

CIRCLE 54 ON READER SERVICE CARD

The first four resemble DOS commands and need no comment. *COPYfile* concatenates two or more data base files, checking for record length compatibility and removing the end of block null characters, whereas in DOS this command is used to copy nondata files as well.

The other two commands in this category are:

```
GETfiles <filename> <filename>...
```

to make data base files active, and

```
DISK <filename>
```

to send output to a disk file.

Certain commands treat records as their object. The *EDIT <filename.ext>* command enters a general-purpose, full-screen text editor that is most often used to write NPL program files. It is also used to make changes to NPL master files in which each record describes a field in a corresponding data file. An alternative to this editor is to call your favorite word processor, as in *DO\WS* to call WordStar from the DOS root directory.

The *CREATE <filename>* command calls a special mode for creating the records of a master file. Before leaving this mode you can edit existing records; afterwards, use the *EDIT* mode.

The *MODIFY <filename>* command followed by the *RUN* command is used for direct viewing and editing of a data base file's records.

Index files are the objects of the index and select commands. The *INDEX <fieldname>* command builds an index table that is used by the system to search for and sort records.

The following command activates a particular index table and specifies which values are to be included in subsequent report and update mode requests:

```
SELECT <indexed fieldname> =
   <value> <value>...
   <value range>...
```

The following statements take program variables as their object. The statement:

```
SET <name>/<format> =
   <expression>.
```

creates and changes program variables from time to time. Up to 255 such variables may be created. These procedural domain variables may include values of nonprocedural domain variables (data file fields or *DEFINE*d variables) with the *LAST* keyword, as in:

```
SET PCT =
   100*(LAST MAX.AMOUNT)
   /(LAST SUM.AMOUNT).
```

where *MAX.AMOUNT* and *SUM.AMOUNT* are computed values from the last line of the preceding report request.

The *SAVE SET TO <filename>* command stores the whole table of program variables in RAM memory to a disk file, which has the file name suffix *SET*.

The *LOAD SET FROM <filename>* command loads a table of program variables from disk file to RAM memory.

Good interactive programs require that a language provide complete flexibility for writing to the computer screen. The simplest command for writing to the screen is an asterisk in column 1 followed by a line of text, so that filling the screen is simply a matter of formatting 24 lines with a leading asterisk. Markers begin-

FTL MODULA-2 $49.95

The most programming power for your money

FTL MODULA-2 gives you:
- Full implementation of MODULA-2.
- Split screen, multi-file editor.
- Sources to the run-time modules.
- Complete support for the product.
- 8087 reals available for MSDOS.*

FTL MODULA-2 meets the standards in Niklaus Wirth's book, "Programming in MODULA-2", third edition. No other language gives you better flexibility, code design, or ease of code maintenance.

FTL MODULA-2 gives you compact, rommable code *quickly*. This is the language of the future. If you program in Pascal or C, FTL MODULA-2 will increase your output per programming hour. If you want to learn high level structured language programming, this is the best starting point.

FTL MODULA-2 was named "product of the year" by Jerry Pournelle in his column, "Computing at Chaos Manor", BYTE Magazine, April, 1986.

* 8087 support is an additionl $39.95

FTL Editor toolkit $39.95
The source code to the editor is offered as a toolkit. The editor was written in FTL MODULA-2 and this set of programs provides the novice with working sources to speed up learning. The experienced programmer will find the wealth of pre-written modules a time-saving bonanza.

SAVE $10.00
Buy both the compiler and the editor toolkit for only $79.95. That's a $10.00 savings off the regular price. **FTL MODULA-2 is available for both MSDOS and CP/M-80 systems.**

WORKMAN & ASSOCIATES ... USER-FRIENDLY SOFTWARE

SEND CHECK OR MONEY ORDER TO:

Workman & Associates
1925 E. Mountain Street
Pasadena, CA 91104
(818) 791-7979 voice.
(818) 791-1013 data M-F 8pm-8am
Or join us on BIX (Byte Information eXchange W.AND.A Conference.)

MasterCard and VISA welcome.
Please add $3.00 Shipping and Handling.
Calif. residents please add 6% sales tax.

CIRCLE 55 ON READER SERVICE CARD

ning with < are used to position data items. For example:

```
* The date is: <DATE
  the time is: <TIME
```

where *DATE* and *TIME* are program variables set by the internal clock.

To paint the screen with various colors, the *COLORBLOCK(b,f,r,c,r,c)* command is used, where the first two integers designate back- and foreground color codes and the other two pairs specify the row and column of two corners of a box on the screen. In Modify mode, *COLORS(b,f)* and *COLOR(b,f)* are available to color the data fields selectively on a data entry screen.

The *MOVECURSOR(r,c)* command positions the cursor, prior to some action by the user. The so-called *Amper* variable is used to generate a prompt to the user and to read the user's response.

Consider a case where MENU.TXT is a file with instructions and a menu of six choices for the user:

```
DISPLAY MENU.TXT
MOVECURSOR(17,40)
SET REPLY/A1 = '&"Enter choice:"[6]'.
CLEAR SCREEN
```

The prompt message appears in the middle of the 17th line, and the system accepts only a number from 1 to 6 and stores it in the program variable *REPLY*. The string of characters *&"Enter your choice:"[6]* is replaced with the number chosen, such as 4, so that upon execution the *SET* statement reads:

```
SET REPLY/A1 = '4'.
```

Other commands, such as input/output statements, are pretty simple. The only question is where to send the results of a report mode request. Results will go to the console screen unless you give the commands *PRINTER*, *DISK <filename>* (to create a report text file), or *AUX* (for the serial port).

To write a new data base file to disk, use the phrase *HOLD AS* (or INTO) *<filename>* in a report request and a data file/master file pair will be created.

Finally, the following are a few of the commands that are used to set certain conditions, clear various stored parameters, and display the values of program variables and file parameters (slashes separate options):

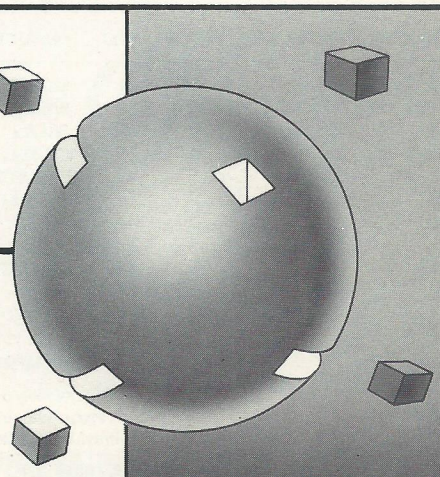
```
HELP ON/OFF
MENU ON/OFF
ECHO ON/OFF/*
DBFILE ON/OFF
FILE?
SET?
CLEAR DEFINE/JOIN/SCREEN/
SELECT/SET/STATUS
```

The power and convenience of NPL are best illustrated by the examples shown in the accompanying figures rather than the details of the syntax. When nonprogrammers can get the desired results quicker than expected, the language will have proven its merit.

NPL was first implemented in 1980-81 on the DEC PDP11 and the Apple II and III computers. The full language is available currently as NPL/R Plus, under MS-DOS on IBM-compatible machines ($595) and the Victor 9000 ($795). A report-writer version for dBASE users with PC compatibles called dNPL/Reporter ($89.50) was announced in February 1987. A smaller version, PC-NPL, is available as shareware on two IBM disks ($20).

Other versions are offered for the DEC Pro 350/380 computers ($695) and the Hewlett-Packard 200/300 series computers ($1,190). Contact the author at Database Applications Inc., (609) 924-2900, for more information.

C PROGRAMMERS!
THE TOOLS YOU NEED
AT A PRICE YOU'LL LIKE



BTree Supports all index file operations. Very quick sequential or random access, duplicate keys, multiple indices, fixed and variable length data records are all supported. **75.00**

ISAM Works on top of BTree to provide a simple, yet powerful application program/file system interface. Complex filesystem manipulation becomes a snap. Provides the power of a database manager with the flexibility of a programming language. **40.00**

lp Finally, a completely device independent printer library! lp drives any printer as accurately as possible and allows easy access to its most sophisticated features. Multiple fonts, multi-column output, complex margin formatting, and much more. Pays for itself the first time it's used. **75.00**

Snake The ultimate 'make' utility. We couldn't find a good one, so we wrote a great one. Has all kinds of powerful features including wild card filename expansion, nested macros, and multiple dependency and rules definitions. Ready to go for MS-DOS; C source is there if you use another operating system. **59.00**

Combine & Save: BTree + ISAM + lp **149.00** + Snake **199.00**

Each product includes a typeset manual, example programs, and complete C source code that runs on any operating system. Softfocus products may be incorporated into applications royalty-free.

Credit card orders accepted. Visa, M/C, Amex. Dealer inquiries invited.

softfocus
1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

CIRCLE 56 ON READER SERVICE CARD

References

1. Martin, James. *Application Development Without Programmers*, Prentice-Hall, 1982.
2. Truitt, T.D. and S.B. Mindlin. *An Introduction to Nonprocedural Languages Using NPL*, McGraw-Hill, 1983.

Thomas D. Truitt is President of Database Applications Inc. and a designer of the NPL system. He has designed many large data base systems using nonprocedural systems on 370 class computers. He holds an MSE degree in Electrical Engineering from Princeton University in Princeton, N.J. and wrote his first compiler in 1958.


```
List full name, but sort by last name and then by first
name. Reformat the salary column with commas and $

DEFINE FULLNAME/A30 = CLIP(FIRST)+' '+CLIP(MI)+' '+LASTNAME
       PAYRATE/D11CR = SALARY.

PRINT FULLNAME DEPARTMENT PHONE PAYRATE AS Salary,
   IF DEPARTMENT IS SALES OR MARKETNG
   BY LASTNAME NOPRINT, BY FIRST NOPRINT.

FULLNAME              DEPARTMENT PHONE              Salary
--------              ---------- -----              ------
Nathan S. Adams       SALES      (555)633-0937      $15,675
Debbie S. Anderson    MARKETNG   (555)558-2674      $33,633
Kim L. Arlich         MANUFCTR   (555)624-8773      $15,775
Sandy B. Beman        MANUFCTR   (555)680-0554      $18,063
    .
    .
```

Figure 5.

```
A simple listing of employee information, in the
sequence of the file itself.

PRINT REC# FIRST LASTNAME PHONE DEPARTMENT.

 REC  FIRSTNAME   LASTNAME    PHONE            DEPARTMENT
 ---  ---------   --------    -----            ----------
   1  Nathan      Adams       (555)633-0937    SALES
   2  Debbie      Anderson    (555)558-2674    MARKETNG
   3  Kim         Arlich      (555)624-8773    MANUFCTR
   4  Sandy       Beman       (555)680-0554    MANUFCTR
        .
        .
```

Figure 2.

```
Sort by employee last name, z thru a

BY HIGHEST LASTNAME, PRINT FIRST ROW# REC# DEPARTMENT SALARY.

LASTNAME     FIRSTNAME   ROW  REC   DEPARTMENT  SALARY
--------     ---------   ---  ---   ----------  ------
Zambini      Rick        1    31    SALES       51876
Youngblood   Dick        2    50    SALES       32302
Young        Sandy       3    39    SALES       21852
Skyhawker    Luke        4    18    ENGNRG      38918
Sanders      Kathy       5    47    MANUFCTR    38168
Rodan        Bill        6    26    ENGNRG      35087
Rizzo        Ann         7    44    SALES       14575
             Anthony     8    35    SALES       19367
Rivera       Harry       9    23    MANUFCTR    38714
Pope         Jan         10   42    SALES       40857
    .
    .
    .
```

Figure 3.

```
Write a two table of employee statistics.
Limit salary field width to 7 characters.

DEFINE PAYRATE/D7C = SALARY.

WRITE CNT.EMPID# AS NBR AVE.AGE AVE.PAYRATE
  MAX.PAYRATE MIN.PAYRATE, ACROSS SEX, BY DEPARTMENT.

           SEX
           F                                  M
                AVE AVE      MAX      MIN          AVE AVE      MAX      MIN
DEPARTMENT NBR  AGE PAYRATE  PAYRATE  PAYRATE  NBR  AGE PAYRATE  PAYRATE  PAYRATE
---------- --- --- -------  -------  -------  --- --- -------  -------  -------
ADMIN        1  47  27,600   27,600   27,600    2  51  46,146   59,000   33,291
ENGNRG       .   .       .        .        .   10  45  36,310   42,745   30,849
MANUFCTR     8  38  22,594   38,168   15,775    4  38  22,955   38,714   17,475
MARKETNG     3  35  34,539   53,684   16,300    .   .       .        .        .
SALES        9  32  28,952   54,152   14,575   14  36  27,747   56,444   15,675
```

Figure 4.

```
Show ADMIN employee data on two lines;skip a line between employees

PRINT FULLNAME PHONE PAYRATE AS '......Salary :'
 OVER DEPARTMENT EDYRS AS Educ: DEPENDENTS AS '    #Deps:'
       EMPID AS Emp_id#: BOSSID AS 'Boss id#:', SKIP,
IF DEPARTMENT IS ADMIN,
BY LASTNAME NOPRINT BY FIRST NOPRINT.

Robert T. Hamilton          (555)760-8959 ......Salary :     $59,000
ADMIN    Educ:  18  #Deps:    2 Emp_id#:   4792 Boss id#:   4792

Mike H. Johnson             (555)352-3445 ......Salary :     $33,291
ADMIN    Educ:  16  #Deps:    2 Emp_id#:   4888 Boss id#:   4792
```

Figure 6.

```
Print mailing labels, for the state of Oregon.
Use prior definition of lastname; define citystzp.

DEFINE CITYSTZP/A30 = CLIP(CITY)+', '+STATE+' '+ZIPCODE.
PRINT FULLNAME OVER STREET OVER CITYSTZP, SKIP 1, BY ZIP
  NOPRINT, NOPAGE IF STATE IS OR.

Jan B. Pope
7100 Fulton Pl.
Beaverton, OR 97007

Sara P. Collins
14234 Riverside Dr.
Portland, OR 97219
```

Figure 7.

Saywhat?!™

Now it's easy to design screens in any computer language.

Saywhat?!™ is the newest tool for software designers and programmers. It's the fastest way to create stunningly beautiful screens for programs, prototyping, and slide shows. Saywhat?! is so intuitive to use that even people who have never created a screen before will feel right at home. Yet it has all the performance that veteran programmers want.

Speed!

Powerful commands mean *fast* screen design. Create screens quickly and visually in a "what you see is what you get" environment. Then, with one simple command in your program, your screens will all pop up like *magic*. Any program you write will run faster and look better with Saywhat?! screens.

Features!

Everything you expect in a professional screen design tool is right at your fingertips, whether you use a color or monochrome system.

- Multi-style lines
- Color and monochrome drawing
- Walk-about mode
- Automatic box generation
- Block copy
- Rubber stamping
- Block paint
- Block move
- Block erase
- All color combinations
- 16 border colors (CGA)
- Text entry
- Blinking
- Extended character set
- Undo
- Print screen
- Direct access to DOS.

The endless possibilities make beautiful screen design easy and fun.

Windows!

All your programs, even batch files, can have windows and instant help screens, up to three levels deep. Now you can give your programs that crisp "Sidekick™" feel.

Languages!

Thanks to our new resident utility, instant screens are possible with any language running under PC/MS-DOS™. Like dBASE, Pascal, BASIC, C, Modula-2, FORTRAN, and COBOL. Saywhat?! is the perfect companion for the Clipper, WordTech, FoxBase, and QuickBASIC compilers. And screens can be modified without recompiling your code. Saywhat?! even writes clear, concise data entry code for dBASE, Turbo Pascal and BASIC. Slide shows and program prototypes can be arranged with as little as a simple batch file.

Value!

Now comes the best part—a sensible, even incredible, price. $49.95. That includes a 68-page bound manual, a quick reference guide, sample screens, eleven utilities, and technical support. Not copy-protected. No licensing fees.

Just call the toll-free number to order Saywhat?! today.

When it comes to guaranteed, lightning-fast, first class screens, you're no longer on foreign ground.

Saywhat?!

$49.95

(+ $5 shipping and handling)

800-HOT-WARE
(800-468-9273)

In California:
800-231-7849

In Canada:
800-663-9361

International:
415-571-5019

The Research Group
88 South Linden Ave.
South San Francisco
CA 94080

Saywhat?! runs on IBM PC/XT/AT™ and compatibles. Saywhat?! is a trademark of The Research Group. Sidekick and Turbo Pascal are trademarks of Borland International, Inc. IBM PC/XT/AT and PC-DOS are trademarks of International Business Machines, Inc. MS-DOS is a trademark of Microsoft Corp. dBASE is a registered trademark of Ashton-Tate, Inc.

© 1987 The Research Group

CIRCLE 57 ON READER SERVICE CARD

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

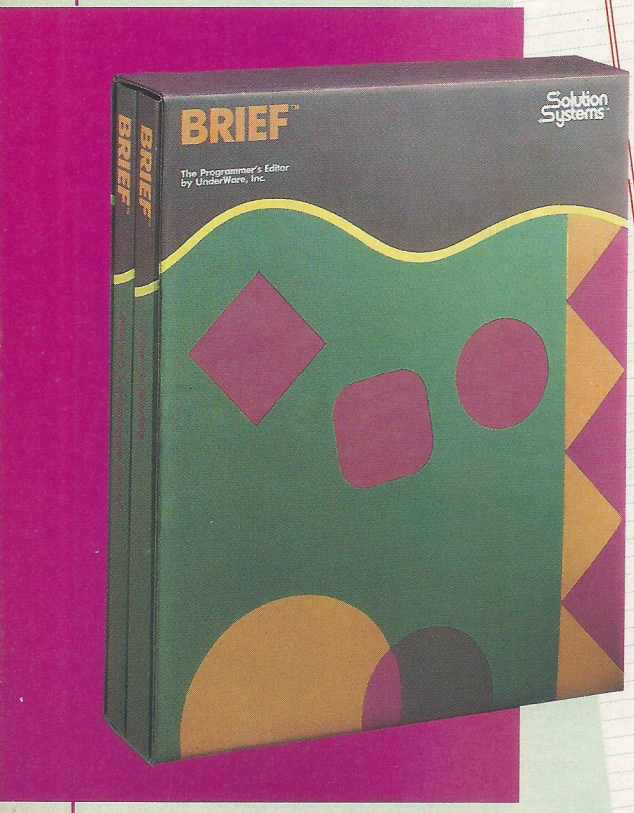
Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "**Recommended. If you need a general purpose PC programming editor, look no further.**" His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.**

TO ORDER CALL: 1-800-821-2492 (in MA call 617-659-1571)

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

Solution Systems™
335 Washington St.
Norwell, MA 02061
(617) 659-1571



Look at these BRIEF 2.0 enhancements!

Main Features:

- *All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.*
- *Setup program for easy installation and configuration. (Requires no knowledge of the macro language)*
- *Increased speed for sophisticated operations like Undo and Regular Expression Search.*
- *Expanded regular expressions, with matching over line boundaries.*
- *More block types, with marking by character, line or column.*
- *Command line editing (move cursor, add and delete characters, specify command parameters).*
- *Support for more programming languages.*
- *Optional borderless windows.*
- *Enhanced large display support, including wider displays.*
- *Reconfigurable indenting for C files (supports most indenting styles).*

*Plus the basic features that made BRIEF **SO** popular!*

Basic Features:

- *Full multi-level Undo*
- *Windows*
- *Edit many files at once*
- *File size limited only by disk space*
- *Automatic language sensitive indentation*

Requires an IBM PC or compatible with at least 192K RAM.

BRIEF is a trademark of UnderWare, Inc.
Solution Systems is a trademark of Solution Systems.

CIRCLE 58 ON READER SERVICE CARD

PRODUCT BINGO

By Joanna M. Poole

Each month Product Bingo features the latest in new software and hardware products of interest to COMPUTER LANGUAGE *readers. Send new product information to Regina Starr Ridley, Editor,* COMPUTER LANGUAGE, *500 Howard St., San Francisco, Calif. 94105.*

Turbo power to the people

TurboPower Software's *Turbo OPTIMIZER* increases the speed of and condenses Turbo Pascal programs and sells for $75.

TurboPower Technical Productivity Software, 3109 Scotts Valley Dr., Ste. 122, Scotts Valley, Calif. 95066, (408) 438-8608

CIRCLE 120 ON READER SERVICE CARD

Frustrated?

Help is on the way with *TECH Help!–The Electronic Manual* from *Flambeaux Software*, an on-screen technical reference manual for system-level programmers that includes coverage of DOS and ROM-BIOS services. $69.95

Flambeaux Software, 1147 East Broadway, Ste. 56, Glendale, Calif. 91205, (818) 500-0044.

CIRCLE 121 ON READER SERVICE CARD

Mothering from Mylex

The *Mylex 386-A* motherboard utilizes the Intel 80386 microprocessor and runs at speeds up to 16 megahertz, with a Norton rating of 23.5.

Mylex Corp., 5217 N.W. 79th Ave., Miami, Fla. 33166, (305) 592-9669

CIRCLE 122 ON READER SERVICE CARD

Turbo tools

Dig in with *Mach 2*, a tool chest of assembler and Turbo subroutines and functions now available from *MicroHelp* for $69.

MicroHelp Inc., 2220 Carlyle Dr., Marietta, Ga. 30062, (800) 922-3383

CIRCLE 123 ON READER SERVICE CARD

A boss that does windows?

Star Guidance Consulting announces its window function library for C, *The Window BOSS*. The function library supports C compilers from Lattice, Microsoft, Computer Innovations, Datalight, Borland International, and Mark Williams. $50 includes source code

Star Guidance Consulting Inc., 273 Windy Dr., Waterbury, Conn. 06705, (203) 574-2449

CIRCLE 124 ON READER SERVICE CARD

Second time around

Spruce Technology has rewritten in C the latest version of its syntax-directed editors, *FirsTime v. 2.0*.

Spruce Technology Corp., P.O. Box 7948, Shrewbury, N.J., (201) 741-8188

CIRCLE 125 ON READER SERVICE CARD

Cornucopia

The KORN shell (KSH), offers system and software developers the functionality of the Bourne Shell programming language with the C Shell Interactive facility. $195 for the AT&T 3B2 and $125 for Xenix and Microport versions.

ASPEN Technologies Inc., P.O. Box 5727, Parsippany, N.J. 07054, (201) 316-0866.

CIRCLE 126 ON READER SERVICE CARD

New pick for the Mac

A full ANSI '77 FORTRAN development system for the Macintosh is available from *DCM data products*. *MACTRAN77* is integrated with an editor, compiler, linker, and symbolic debugger.

DCM data products, 1710, Two Tandy Center, Fort Worth, Texas 76102, (817) 870-2202

CIRCLE 127 ON READER SERVICE CARD

Interactive aids

InterTools is *Intermetrics*' line of software development tools for programming embedded microprocessors supported on VAX, Micro VAX, VAXstation II, Sun, Apollo, and IBM PC/XT/AT.

Intermetrics Inc., 733 Concord Ave., Cambridge, Mass. 02138, (617) 661-0072

CIRCLE 128 ON READER SERVICE CARD

CLARION translator

Barrington Systems announces the *CLARION translator*, a compiler module for generating .EXE files directly executable from DOS. $95

Barrington Systems Inc., 150 Sample Rd., Pompano Beach, Fla. 33064, (305) 785-4555

CIRCLE 129 ON READER SERVICE CARD

More translations...

LTS_2_C from *Resource Analysis International* is designed to translate any spreadsheet that conforms to the WK1 and WKS file formats into C source files.

Resource Analysis International Inc., 12581 Venice Blvd., Ste. 204, Los Angeles, Calif. 90066, (213) 390-7661

CIRCLE 130 ON READER SERVICE CARD

C the Advantage.

Now, with MS Windows Support!

There's never been a better time to buy Lattice C. Professional programmers the world over have made Lattice C the standard compiler for serious MS-DOS programming. Now Version 3 offers even more of the features that have made our previous versions so popular. Our new compiler features include:

ANSI language constructs including, *unsigned* as a modifier, *void* data type, *enum* data type, structure assignments, structure arguments, structure returns, and argument type checking.

The compiler also contains better aliasing algorithms, more efficient code generation, and more flexible segmentation, in-line 8087 code generation, and 80186/80286 code generation.

The library contains more than 200 new functions, including: ANSI/UNIX/XENIX compatibility; extended support for MS-DOS; extended support for networking including file sharing, file locking, and I/O redirection; and flexible error handling via user traps and exits. Plus the library has also been re-engineered to produce much smaller executables.

Debugging your Lattice C or Assembly language programs will be even easier when you use our new and improved C-SPRITE™ symbolic debugger.

We've included new features such as a source mode that allows C-SPRITE to work with source lines instead of machine instructions for most functions such as disassemble, single step, and breakpoints. In addition, when you use the source mode, symbol types can now be completely specified so that variables are automatically displayed in their correct data type.

C-SPRITE allows the size of the symbol table to exceed 64k bytes. Symbols can be made to be case sensitive, and when symbols are loaded, compiler-generated symbols can be ignored.

Try the new Version 3 C Compiler and C-SPRITE from Lattice. Because C-ing is believing.

Lattice

Lattice, Incorporated
P.O. Box 3072
Glen Ellyn, IL 60138
312/858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES: Benelux: Ines Datacom (32) 2-720-51-61
Japan: Lifeboat Inc. (03)293-4711 England: Roundhill (0672)54675
France: SFL (1)46-66-11-55 Germany: (49)7841/4500 (49)8946/13290

CIRCLE 92 ON READER SERVICE CARD

SOFTWARE REVIEWS

Tools of the trade

For almost every project a programmer handles for a customer, company, or him or herself, the most important criterion of success is speed. No, not the milliseconds saved with that neat bit shift, but speed as in "When are you going to finish that thing?"

After speed comes time: your time. Maybe it's a set-fee project; the more projects you complete, the more you make. Or maybe you're on salary, and the only things on the line are your reputation and career. (The little things.)

Then there's elegance and durability. If you can use extant code to handle mundane chores, you'll have more time to devote to well-designed solutions that will stand up to modification as the system grows and changes.

It always comes back to that most precious of commodities, time. With the right tools, you can get the job done more quickly and hurry home to your true love: the 3-D version of Pac Man that's going to make you that million.

The time saved by using the right tools belongs to others as well. The real goal of any project is more than just getting the thing off your desk—it's increased user efficiency. For most applications programs, the chief benefit of custom design is the savings in operator time of an easy-to-use package that's well tailored in both overall design and user interface. The biggest cost for any program is the time spent developing, learning, and using it.

These five wrap-ups cover two tools of the trade that can be a real boon to the software professional: prototypers and screen managers. Prototypers can help set the project specification and speed-up design; screen managers jump in at a later stage, handling keyboard and screen details. With the two, your product should be finished sooner, look better, and operate more smoothly.

As always, the products available reflect vendors' perceptions of the size of the potential market. We've split the products up into four prototypers and nine screen managers that support C, four Turbo Pascal prototypers/screen managers, three COBOL screen managers, and a couple curses packages thrown in for good measure.

With a prototyper, users can pick from a range of possible designs, occasionally resulting in a semifinal version of a program specification. The prototypers handle text fields, input and display fields, graphics, and cursor control. Code generated by the prototypers can be used as a working model or a superstructure to build the application around.

The screen managers' features vary, and what you'll need will depend on what you want to do. A number of products are available to handle pop-up windows and help screens. We chose to focus on screen managers that handle data entry and validation.

The Turbo Pascal packages offer both prototyping and screen management in a single package, reflecting that language implementation's focus on development speed. The COBOL products are strictly screen managers.

The final wrap-up covers curses, a library of screen management routines for controlling multiple windows and screen buffers. Curses is now available on the PC and should be of interest to those who program on PCs and in a UNIX environment on a larger machines.

Program prototypers and the illusion of reality

By John McNamee and Ken Thompson

Have you ever wished you could see how an application would look before you started coding? Have your clients wanted to see a demo before they approved a major project?

There are many reasons for doing a prototype before starting work on the real thing. But until recently, it could take almost as long to build a prototype as the final program.

With this wrap-up, we'll cover four packages that can help make building prototypes fast and easy. The products we'll cover are Software Garden's Dan Bricklin's Demo Program, Nostradamus's Instant Replay, Genesis Data Systems' Screen Master, and Skylight Software's SKYLIGHTS.

Each of the tools has a somewhat different focus. Dan Bricklin's Demo Program creates vaporware and slide shows. Instant Replay is optimized for demos of existing or semiworking programs. Screen Master includes libraries to integrate prototype screens into a final program. SKYLIGHTS has support for a mouse and graphics.

All of the packages require an IBM PC or compatible with 256K RAM. None require a hard disk, but all are easier with one. Only SKYLIGHTS supports a graphics mode; all others only work with text screens.

Software Garden Dan Bricklin's Demo Program

Dan Bricklin's Demo Program from Software Garden allows applications to

be prototyped without writing any code. A prototype may be vaporware or a slide show. (For more on the design of this product, see "Entrepreneurship the Dan Bricklin Way," *COMPUTER LANGUAGE,* Dec. 1986, pp. 101-104.)

Vaporware is a prototype of a program that doesn't currently exist. When you want to see how a new program might work before coding it, you create vaporware. Vaporware is interactive, allowing the user to make choices about program flow and to simulate data entry.

Slide shows are a series of screen images displayed one after the other in a predefined order and at a set speed. There is no user interaction. Examples of slide shows are the self-running demos often seen at computer stores and trade shows.

In Demo, vaporware and slide shows are called (not surprisingly) demos. Demos are organized into a series of slides that are essentially screen images. A slide may have other slides overlaid on top of it (null characters are transparent, allowing data from lower slides to show through) and run handlers associated with it.

Run handlers control user interaction with a demo. They specify a series of conditions and actions. Conditions are such things as keys being pressed and time-outs. Actions include generating tones, displaying different slides, and stopping the demo.

Demo is not just a collection of tools but an integrated environment. As many slides as memory will hold are stored in one file, and Demo itself is a single program. This approach eliminates keeping track of a lot of file names, and you don't have to switch between programs to do different tasks.

We liked a lot of things about the Demo user interface, especially the ability to use function keys or menu selections. Infrequent or novice users can do everything with menu selections, while more advanced users can use function keys. Menus only appear when you call for them and automatically adjust their position so as not to obscure the area around the cursor.

Demo's general screen-editing interface is intuitive. The line-drawing facility is a good example. All packages do line drawing, but Demo understands corners and intersections between different line styles. For example, if you draw a single line that crosses over a double line, Demo will choose the appropriate character for where they intersect.

Demo includes a resident utility program to capture screens from existing applications. Captured screens are buffered in memory rather than written to disk, limiting the number of screens that can be captured before Demo must be run to import them.

The one major feature missing from Demo is assistance in creating menus. Demo can prototype any sort of menu scheme but offers no assistance in creating standard forms (for example, Lotus 1-2-3 style). Menus must be built one slide at a time with overlays helping to reduce duplication. Run handlers must be created for each menu selection—a repetitive process that the computer should handle for you. Less importantly, attributes must be entered in hex, which is difficult even for experienced programmers.

Demo comes with a 29-page reference manual; a 96-page tutorial manual and disk is an extra $49.95. The reference manual is printed on 8½x11-inch paper with a dot-matrix printer; a binder isn't included. Selling tutorials separately and not including a binder may have been OK when only hackers used computers, but it's unacceptable today.

All in all, we really liked Demo. It has an excellent screen editor, and the integration between editing and prototyping features is unmatched by any other product. The only feature really missing is assistance in creating menus.

Nostradamus Instant Replay

Instant Replay from Nostradamus is a tool for creating demonstrations, tutorials, and prototypes. This product is clearly aimed at people who want to create demos or on-line tutorials of existing programs; prototyping new applications seems to have been included as an afterthought.

During the wrap-up, we worked with a beta copy of Instant Replay v.2, which should be available by press time. Owners of v.1 may upgrade for $50. This review will focus on Instant Replay's prototyping features. (For discussion of the product's demo and tutorial capabilities, see the review of Instant Replay, *COMPUTER LANGUAGE*, Jan. 1987, pp. 128-132).

Rather than being a single, integrated program, Instant Replay is organized as an interconnected set of tools. A main menu is used to link the various modules; infrequent users may have trouble remembering how it all fits together.

The Screen Genie module is responsible for screen editing. It allows you to create screens in a reasonably intuitive manner. Commands are entered with function keys, which we found a stumbling block for novice users. On-line help lists what all the keys do.

Screen Genie has all the usual screen editor features that allow such things as controlling attributes, entering special IBM graphics characters, and drawing lines and boxes. The line-drawing function understands corners but doesn't handle intersecting lines.

Each screen is saved as a separate file. Screens may be saved in Screen Genie format (for use by other parts of Instant Replay) or as a .COM or .OBJ file. .COM files can be created so they will run under DOS (the screen image is displayed when executed) or in a programmable format that can be used with Turbo Pascal. For an extra charge, runtime libraries that will display Screen Genie files and handle other user interface tasks are available.

Vaporware and slide shows are built with the Proto Typer program; Screen Genie creates the screens and windows that make up a prototype. You must switch back and forth between the two programs as a prototype is developed.

Slide shows are built from a set of sequentially numbered screens. To create a slide show, you specify a base file name (which will have a slide number appended to it) and the keys used to increment and decrement the slide number (Page Up and Page Down are usually used).

Interactive vaporware is built by specifying an initial screen and a series of hot keys with associated actions. When an interactive prototype is run, the initial screen is displayed, and the system waits for one of the hot keys to be pressed. Possible actions include displaying another screen, popping up a window on top of the current screen, processing a menu, or running another prototype.

Proto Typer has special support for menus. A menu is created with Screen Genie just like any other screen, except menu items have brackets around them. The brackets can be hidden from display by setting their screen attributes to black on black.

When a menu is called from inside a prototype, the menu screen is displayed and a menu bar can be bounced between the possible selections using the arrow keys. A different screen or prototype is activated depending on which menu item is selected. (The menu item number is added to a base file name, similar to the way slide shows are handled.)

Instant Replay has many more features we haven't talked about. In fact, it has more features than any one person probably knows what to do with. Having all that power is nice when you need it, but it gets in the way during simpler tasks. Whether Instant Replay is right for you depends a lot on whether you need all it has to offer.

Genesis Data Systems
Screen Master

Screen Master is a screen design tool coupled with a prototyper and subroutine library. It includes a resident utility that captures screens from existing programs. The package is aimed at Microsoft BASIC and Pascal programmers who want to prototype a new application and then integrate prototype screens into the final program.

The Screen Master editor works with screens and objects. A screen is a 25-line-by-80-column image, and an object is anything less that that, from a single character up to one character less than a full screen. Each screen or object must be saved in a separate file. If a prototype is being created, the file names must consist of a base name plus a sequence number (FOO1, FOO2, and FOO3, for example). It is up to the user to assign and keep track of sequence numbers.

The screen editor is not very intuitive. All top-level operations are invoked with function keys. Novice or infrequent users will have to keep the manual near by or constantly refer to the single on-line help screen that lists each function key.

The insert and delete commands work differently than any screen editor we've ever seen. When a character is inserted, every character to the right of the insert position and every character on every line under the insert position is moved one position to the right. The character at the bottom right of the screen will be lost. Deleting characters works in just the opposite way, moving everything left one position.

This method of inserting and deleting characters was not what we expected. In addition to being counterintuitive, it's hard to use. The manual even warns that using insert and delete "will require practice."

Marking objects and drawing boxes is harder than it should be. Screen Master requires the user to position the cursor at the upper left corner of a box, press a key, move the cursor to the lower right corner of the box, and press a key again. While the cursor is moving, there is no visual indication of what the final size of the box will be. Most other screen editors use rubber-band boxes, where the outline of a box can be stretched, compressed, and moved with the arrow keys until it looks right.

Screens can be captured from existing programs using Magikey, a memory-resident program with several unfortunate flaws that limit its utility. Magikey can only capture a screen when a program is waiting for input. The hot key used to initiate a capture is the tilde (a valid ASCII character). There is no way to abort once you have pressed the hot key.

At last! - Fast, On-screen

FLOWCHARTS

Finally! An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work.

Interactive EasyFlow is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With this program you can quickly compose charts on the screen. More important, you can easily modify charts so they are always up to date.

Features: • Text is automatically centered, character by character, within shapes as you type it • Text formatting controls allow you to over-ride the automatic formating where desired • Lines are created by specifying the starting and ending points - the program automatically generates the route • Cut and paste facility allows arbitrary chart fragments to be moved, copied rotated, reflected or sent to/from disk • Shape insert-delete and row/column insert-delete • Charts can be up to 417 characters wide by 225 lines high. Charts too wide for the printer are automatically printed in strips. • Charts can be larger than the screen - the window into the chart scrolls both horizontally and vertically as necessary • Works with many popular matrix printers including Epson, IBM graphics printer and compatibles. Full support for HP LaserJet and LaserJet Plus. Works with HP 7475A (& compatible) plotters. Can be used with ANY printer when non-graphic (character) output is acceptable • All standard flowcharting shapes included • Most shapes supplied in large, medium and small sizes • Extensive manual (100+ pages) includes many examples • Context sensitive "help" facility provides immediate assistance at any time • Any number of titles can be placed on a chart • Commentary text blocks can be placed anywhere in the chart • Fast: written in assembly language • Plus many more features than we can mention here

Requires at least 320K memory, DOS-2 or higher and an IBM or Hercules compatible graphics card. On EGA, full 640x350 resolution is used.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by MO, check, VISA, MasterCard, COD or Company PO. Rush orders accepted ($15.00 S&H; USA/Canada only). Rush orders received by noon will be delivered the next business day (to most locations).

Order Desk: 1-800-267-0668

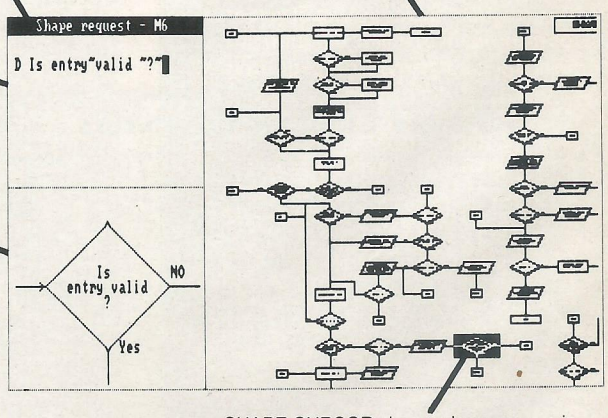
The sample screen display shown below is typical of what you see while editing a chart. Other screen displays are provided for entering titles, changing options, getting "help" and so on.

STATUS BAR (not to be confused with a wet bar) tells you what Interactive EasyFlow is doing at all times.

CHART WINDOW gives an overview of your chart; this example shows the "normal" view. "Close-up" view shows a smaller part of the chart in more detail. "Wide-angle" view shows a larger part of the chart at reduced size.

TEXT/MESSAGE WINDOW used to enter user text and to display messages from Interactive EasyFlow.

CURRENT SHAPE WINDOW - shows the content of the current flowchart shape (the one under the SHAPE CURSOR) in complete detail.



SHAPE CURSOR shows where you are in the chart. Cursor keys move it around; chart window scrolls if you run off the edge of the window.

HavenTree Software Limited
P.O. Box 1093-K
Thousand Island Park, NY 13692
Information: (613)544-6035 ext 46

CIRCLE 60 ON READER SERVICE CARD

The Screen Diemon (pronounced "demon") program is used to connect a series of screens and objects to produce slide shows or vaporware. Screen Diemon allows for conditional branching to other screens based on keyboard input, can loop over a set of screens and objects, and handles automatic animation.

You must do considerable planning before using Screen Diemon; it helps to sketch out on paper how the final prototype will flow. Also, the general requirement that file names include a sequential number is irritating.

The documentation is IBM-sized; it doesn't include a binder. The manual is organized into sections for each module, with both tutorial and reference sections. Some examples on using the run-time library from Pascal should have been included.

Overall, we weren't impressed with Screen Master. The package is outdated, lacks features, and is difficult to use.

Skylight Software SKYLIGHTS

SKYLIGHTS is a sophisticated application generator that includes prototyping. C programmers can create window-oriented applications in text or graphics modes using a wide variety of display and pointing devices. (We were unable to test the graphics capabilities.)

More so than any other product reviewed here, SKYLIGHTS forces a programmer to adopt its user-interface method. If your application doesn't fit into the SKYLIGHTS mold, this package is not for you. On the other hand, if you like the SKYLIGHTS method, the package will automate many user interface tasks that would ordinarily be difficult to code.

SKYLIGHTS includes drivers for a wide variety of pointing devices and display cards. SKYLIGHTS drivers are small resident programs that must be loaded before using the package. Strangely, while drivers are included for a number of exotic display boards, there is no driver for the popular Hercules Graphics Adapter.

Pointing devices include mice, light pens, graphics tablets, and the keyboard arrow keys, but you'll need a mouse or similar device to easily use SKYLIGHTS. I tested it with an optical Mouse Systems PC Mouse that works only when used on top of a special pad. The SKYLIGHTS mouse driver had trouble scaling mouse movements to cursor movements. Moving the mouse all the way across its pad did not move the cursor all the way across the screen—we had to pick up the mouse to move the cursor all over the screen.

SKYLIGHTS has two major components: the editor and the run-time C library. Also included are a demo maker, screen capture program, and report generator.

The editor is used to create windows; up to 64 windows can be saved in a catalog file. Each window consists of a picture and touch zones, which we'll discuss later.

The editor's user interface is unique. Once you understand it, the interface is consistent and easy to use, but getting started is difficult. No on-line help is available, and the manual is hard to follow in places. A number of key concepts didn't become clear until we had actually used the editor for a while.

The screen editor has all the usual features, including box and line drawing, cut and paste, and attribute selection. The mouse makes selecting options and marking areas of the screen a simple task compared to packages that use function keys.

The run-time library has functions to display windows created with the editor and to handle touch zones. Windows can overlap each other and be moved around the screen. Functions are also included to output to windows, draw lines and boxes, read the keyboard, and generate sound.

Touch zones are areas of a window that can be touched with the pointing device. A user-defined function is called whenever a zone is touched. Each zone may have its own function, or one function may handle several zones. The editor is used to define the size and location of each zone and specify the function to be called. Touch zones can be grouped together to form menus.

Using touch zones requires an understanding of event-driven programming environments. Unlike the usual polling environment, an event-driven program does not explicitly request input. Instead, it waits for the system to inform it when an event has occurred.

Events are like hardware interrupts, only at a higher level. For example, when the user clicks the mouse button with the cursor over a touch zone, an event is generated and a user-defined function called.

Programming in an event-driven environment requires a different approach than programming in a polled environment. It can be hard to grasp at first, but once understood it is a clean and efficient way of handling input.

Prototypes can be created using the editor and Demo Maker. Demo Maker uses the SKYLIGHTS touch zone facility to link windows together. The specification of each touch zone includes the name of another window to open if that zone is touched. Menus and pop-up windows can be created by selecting the appropriate options when creating a touch zone. The source code for the Demo Maker is included, so additional features can be added easily.

The documentation is a bit unclear in places and lacks an installation guide. The information on loading drivers is especially difficult to follow. The graphics version of SKYLIGHTS comes with the text version manual and a supplement describing the differences between the two versions.

SKYLIGHTS is an excellent tool for building applications that fit into its mold. If your application doesn't, this product is not for you. ■

John McNamee is the senior software engineer at Genoa Technology, Los Angeles, Calif., where he works on automated quality assurance systems.

Ken Thompson is president of MicroWizards, a PC consulting firm in Los Angeles, specializing in systems-level progammer productivity tools.

Prototypers and manufacturers

Dan Bricklin's Demo Program—$74.95
Software Garden Inc.
P.O. Box 373
Newton Highlands, Mass. 02161
(800) 225-5800, ext. 8088

Instant Replay—$149.95
Nostradamus Inc.
3191 S. Valley St.
Ste. 252
Salt Lake City, Utah 84109
(801) 487-9662

Screen Master—$99.95
Genesis Data Systems
5403 Jonestown Rd.
Harrisburg, Pa. 17112
(717) 652-1200

SKYLIGHTS
Text version—$395
Graphics version—$545
Skylight Software Inc.
2 Charles St.
Bedford, Mass. 01370
(617) 275-2999

HOT GRAPHICS PACKAGE FOR C PROGRAMS.* $39.95

Everything you need to write dramatic graphics effects into your Eco-C88 C programs. Some of the features include:

- Support for EGA, CGA, and Z100
- Over 100 graphics and support functions, many of which are PLOT-10 compatible.
- Many low level support routines reside outside your small model code-data area
- Can write dots thru the BIOS (for compatibility) or to memory (for speed)
- Graphics function help from CED editor available
- World, pixel or turtle color graphics modes
- 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash and fonts
- Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual.

A must for the graphics enthusiast and a bargain at only $39.95

*Requires Eco-C88 C Compiler.

NEW POP-UP WINDOWS FOR YOUR C PROGRAMS.

This windowing library allows you to add pop up windows in your C programs quickly and easily. Use them for help windows, selection menus, error messages, special effects—anywhere you need an attention getter. Just some of the features include:

- CGA, EGA, and monochrome support
- Slow mode option for "flicker" displays
- Control any program that goes through the BIOS
- Use up to 255 windows
- No special window commands; use print f ()
- Resize and move windows
- Custom window titles and borders
- Can be used with ANSI device driver
- Most of window's code-data lies outside small model limits
- Use any of the IBM text or block characters
- User's manual and examples

The Windowing Library requires an IBM PC compatible BIOS and the Eco-C88 C compiler.

ONLY $29.95

HANDY LIBRARIAN MAKES LIFE EASIER.

Now you can combine your modules, functions, and subroutines into your own library for easy link commands. Fully compatible with ANY standard OBJ format files (not just Ecosoft's products). With the Ecosoft librarian, you can:

- Add, delete, and extract from a library
- Get table of contents or index of a library
- Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files.
- Complete with user's manual

A valuable addition for any programmer.

ONLY $29.95

THE FIRST PROFESSIONAL 'C' COMPILER FOR UNDER $60.

A C compiler with many ANSI enhancements at an unbelievably low price. The Eco-C88 C compiler has:

- Prototyping (the new type-checking enhancement)
- Enum and void data types
- Structure passing and assignment
- All operators and data types
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- CC and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime — no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages — enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- CED full-screen program editor

Everything you need at the unbelievable price of $59.95.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later.

Orders only:
1-800-952-0472
Technical Information:
(317) 255-6476

NOT COPY PROTECTED.

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

ORDER FORM CLIP & MAIL TO: Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

ITEM	PRICE	QTY	TOTAL
Flexi-Graph Graphics	$39.95		
Window Library	$29.95		
Eco-Lib Librarian	$29.95		
Eco-C88 C Compiler CED	$59.95		
		SHIPPING	
		TOTAL (IND. RES. ADD 5% TAX)	

PAYMENT: ☐ VISA ☐ MC ☐ AE ☐ CHECK

CARD # _____ EXPIR DATE _____

NAME _____

ADDRESS _____

CITY _____ STATE _____

ZIP _____ PHONE _____

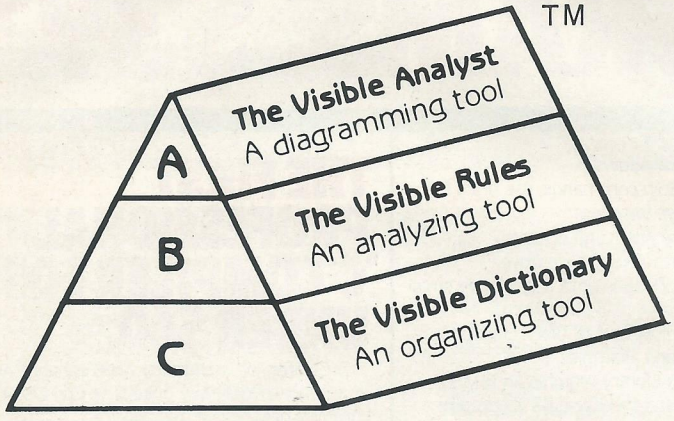
ECOSOFT

CIRCLE 62 ON READER SERVICE CARD

The Only Modular IBM PC Based

SYSTEMS ANALYST WORKBENCH

.....AVAILABLE TODAY



Automation starting at $595.00

Qty = 50 Price $327.25

COMPUTER-AIDED SOFTWARE ENGINEERING .....Your Way

STEP 1 **DIAGRAMMING WITH A FUTURE:** The Visible Analyst - The right place to start for NEW SYSTEM DEVELOPMENT or DOCUMENTING EXISTING SYSTEMS. Features include: mouse oriented graphics, custom symbol/forms generator, tree file structure manipulation with unlimited nested decompositions, variety of line styles including straight, curved, dashed and piped, flexible text processing and editing, object oriented editing. including rubberbanding and block move for FASTBUILD™. **$595.00**

STEP 2 **THE STRUCTURED UPGRADE:** The Visible Rules - Brings Yourdon or Gane & Sarson structured methods into your company. Features - menu selectable expert rules for Yourdon or Gane & Sarson methods, automatic level balancing, six different consistency checks of diagrams or projects, validates project design - output data ASCII Formats provided in 200 page typeset manual with three tutorials. **$595.00**

STEP 3 **THE WORKBENCH UPGRADE:** The Visible Dictionary - Features fast, intelligent and comprehensive dictionary with extensive report capability and automatic population. ASCII File Formats provided for linking. **$595.00**

STEP 4 **THE NETWORK UPGRADE:** The LAN Based Workbench - Currently under development for the Novell Advanced Netware 286 Series of LAN. Features include file sharing and record locking, levels of security and much more. Available Soon. No additional charge for people on our Update/Maintenance program.

STEP 5 **UPDATE/MAINTENANCE PROGRAM:** $50. per Tool per Year. Stay tuned with the EVOLUTION IN SOFTWARE AUTOMATION with our low cost Update/Maintenance program.

COMPARE OUR PURCHASE PRICES AND UPDATE/ MAINTENANCE PRICES WITH OUR COMPETITORS!!!

THEN CALL 617-369-1800

Start Experiencing *THE VISIBLE SOLUTION*™ Today!!!!!

VISIBLE SYSTEMS CORPORATION

49 Lexington St., Newton, MA 02165

CIRCLE 63 ON READER SERVICE CARD

SOFTWARE REVIEWS

Screen managers: The sharper image

By Donald Drumheller and Michael Wilson

More than almost any other program development tool, the features you'll need in a screen manager depend upon the type of application you're building. For this review, we chose to focus on screen managers that handle data entry and validation. While all the products reviewed include more than this relatively complex feature, we believed this to be a good starting point in selecting packages of interest to programming professionals.

We looked at nine packages using the Microsoft C compiler v.4.0: Sydetech System Development Technologies' C-Display Manager, Oakland Group's C-scape, Magus's Data & Windows, Greenleaf Software's Greenleaf Data Windows, Softway's High Screen, Lifeboat's Panel, Creative Programming Consultants' Vitamin C, Vermont Creative Software's Windows for Data, and Data Management Consultants' ZVIEW.

All the packages except one support menu processing and some level of context-sensitive help. Some include the ability to implement editor windows and scrollable forms (for example, a variable number of line items on an invoice entry screen).

Other features include display windows (pop-up or destructive windows for display only) and list-field processing (the ability to cycle through various stored choices for a data-entry field). Further enhancements are full EGA support for screens larger than 25 x 80, automatic support for Microsoft Windows and IBM Topview, and color attribute selection based on monitor type (such as CGA, EGA, monochrome, or monochrome graphics).

We'd like to see all of these features in every package. However, the more features a package implements, the bulkier your application will grow.

Some packages include source code for all modules in the purchase price; others charge extra. Source code is understandably helpful, especially if your application needs some of a package's or compiler's more esoteric functions.

With work, each package could perform or at least mimic many of the cited features already. However, given the complexity of implementing windows and data-entry functions, you may find that a higher-priced package with all the features you need is cheaper in the long run.

When implementing windows and data entry, some of the packages suffer from a severe case of code bloat. One package requires that the large memory model of the compiler be used. The bulkiness of a package's library code can be even more troublesome if calls are made to floating-point routines.

A package written for or compiled under Microsoft C v.4 can be used with Microsoft Pascal v.3.3 or FORTRAN v.3.3 or 4. (You'll still need a copy of Microsoft C v.4 for the libraries, however.) One vendor even provides technical support for interlanguage calling.

A few of the companies have XENIX and UNIX versions that are source-code compatible with the DOS versions. If you anticipate a need for multiuser operation, you may want to consider this feature when choosing a screen generator.

Each package's capabilities and the amount of programming effort required vary greatly. With one package, you can paint your data-entry screen using an external program and just supply the necessary data. When you save your work, you'll have an object file ready for linking to your program. Other packages allow (or force—your choice) the programmer to build the data-entry screen field by field and form by form with function calls.

The product you choose depends on the degree of form processing control you require. Since each package implements its window, data-entry, data-validation, menu-processing, and error-message functions differently, we've opted not to include a features table. Rather than unfairly categorize a product just because it wouldn't fit neatly into a table, we've chosen to detail the design of each package in its respective section.

In general, to implement a data-entry form, a form is defined specifying such things as the window placement, border type, window name, window and field colors, window size, and type of window. Next, you specify the prompt text for each field, a buffer for the resulting value, field type, field placement in the window, a help string, a validation function if desired, and a picture string that defines the allowable characters in each position of the field. Finally, you place the window on screen and read one or more fields. Upon return, your data is in the buffer you specified, ready for further processing.—M.W.

Sydetech System Development Technologies
C-Display Manager

Sydetech's C-Display Manager consists of C-Display Librarian and Display Express (which in turn consists of C-Display Compiler and Demo Express Viewer), all of which are available as separate products. The package has four disks and two manuals.

C-Display Compiler allows the design of screens and data-entry forms. It is a full-screen, page-oriented editor for creating, editing, and reviewing full-screen and window display forms. Colors, attributes, and character-based graphics are supported.

A compilation function translates screen contents and attributes into a data set and saves it as a binary file. This file can then be called by your application program using the run-time support library functions and displayed on the screen. Because of their relatively small size, these screens can be loaded from disk or into memory-resident arrays as required.

C-Display Compiler only creates a display mask; there is no data-entry or validation support at this level. These routines are in the C-Display Librarian package.

Demo Express Viewer has two modules: Demo View and Demo Drive. Demo View reviews screens as if they were in a slide projector. You can set up a presentation for manual operation or with two

levels of timers for continuous demos (one timer is for the entire demo, the other for individual screens). Demo Express Viewer can actually run a demo. Demo Drive's only function is to do just that. Created demos can be distributed with the Demo Drive module without royalties.

C-Display Librarian has five sections: I/O support and video, display management, graphics, and graphics management services.

Video services perform direct BIOS video functions, including controlling the shape, position, and visibility of the cursor. Other functions are for setting and changing colors and screen attributes, printing a string in a specified color and with a specified attribute, printing a character in teletype mode or at the current cursor position, and erasing the entire screen or a limited area (lines or windows).

Display management services control cursor movement, display scrolling, and display editing functions (such as insert and delete). I/O support allows specifying the system environment (current drive, system date and time), data entry and validation, keyboard services (like time-outs and keyboard flushes), loading and saving data files, etc.

Graphics services allow you to draw lines, markers, and boxes and do individual pixel plotting. Also possible is filling in boxes, painting background colors, and erasing the screen. Graphics management is for drawing polygonal lines, circles, ellipses, circular segments, and the like.

Source code is provided for the display management and graphics services and I/O support modules. All functions are included in .LIB files with system- and hardware-dependent files in .H files. The library functions are also included as either C source files or .OBJ files. Video and graphics services are included as object modules.

C-Display Manager is available in two versions: model TX for Texas Instruments machines and model PX for IBM PCs and compatibles. Each model includes two sets of library and object modules: one for programs compiled with Microsoft v.3.0 and one for Lattice C v.2.14 or 3.0. All memory models are supported, and you may distribute your applications without royalties.

The Librarian package handles field types with the I/O support module. Available types include yes/no, ASCII characters (ASCII 32 through 254 and backspace), nonnegative and signed integers, and a file name type (checked for correct syntax only).

You can display any form of menu, but they're for display only. You must use program control for any activities you want to perform. Pictures are not available per se. If you want to change the I/O services source code to accept only certain characters, you can do so, recompile, and link the changed code into your application. C-Display Manager requires a color graphics card (CGA or EGA) for model PX or a Texas Instruments RGB monitor for model TX.

Unfortunately, the majority of any data-entry validation must be supplied by the user. The C-Display Compiler does not support path names, and all in-progress work (screen designs) must reside in the same directory as the utility. The C-Display Editor does not have help screens nor is there a quick reference card—your only recourse while editing is the manual.

C-Display Manager is the only package we reviewed that provided actual graphics capabilities; all others support only text graphics characters. If you want to include bit-mapped graphics in your application, then this might be the package for you.—D.D.

Oakland Group
C-scape

Oakland Group's C-scape has two disks and one 155-page manual. Free source code is provided when you send in your registration card. Support is offered through a Fido bulletin board system and a toll-free 800 number. The bulletin board is worth using; it features beta-test functions the company is soliciting comment on, and examples are also available showing how to use C-scape with Raima Corp.'s db_Vista and Faircom's c-tree.

Screens are defined by library calls only. A replacement for the *printf* function defines the location of text and fields on the screen. You may place as many lines and fields within a screen as available memory (and memory model) will allow.

Once you define your fields, another library function sets the window. Once the window is set, no further changes to a screen are possible (other than data entry). This seems a severe restriction since all the other packages allow you to dynamically change your screen.

All validation is done through a programmer-supplied array of pointers to functions that can be from the C-scape library or programmer written. There is no specific menu or help system in C-scape, but these could be easily implemented. Programmer-supplied functions can be called when the screen is placed and taken down, when entering a field, leaving a field, and on each key press inside a field.

The following compilers are supported in all memory models: Microsoft, Lattice, and Aztec. No royalties are charged to distribute application programs under DOS. UNIX and XENIX versions are reportedly in the works.

C-scape implements only one field type that's equivalent to a string. Since the programmer supplies the entry and exit functions, any sort of input type is allowed. The compiler's floating-point library is called upon only if the programmer uses floating-point operations.

This product does not implement a special menu system. Menus are achieved by using the field entry and exit functions. Other than being slightly more work for the programmer, this method works well—it also means fewer functions to worry about calling properly. C-scape can be used with either a color (CGA or EGA) or monochrome monitor. Colors are achieved by placing a special code in the C-scape version of the *printf* statement for field definition.

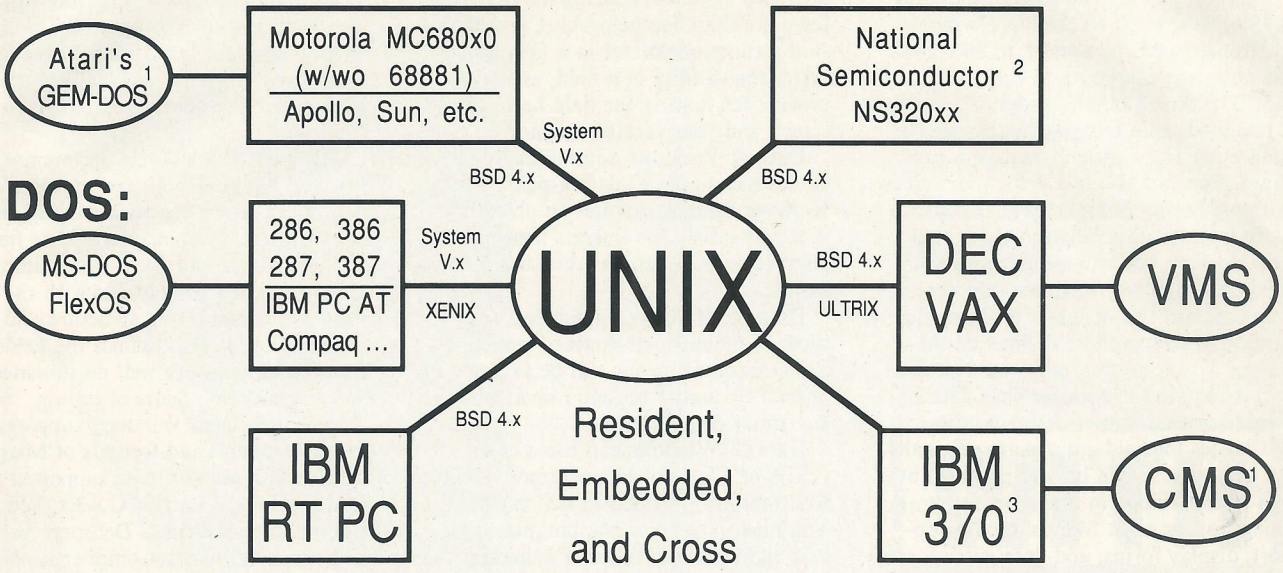
C-scape is somewhat light on features. It has no editor windows or predefined, working help systems, although editor windows are to be included in v.2.0. Fortunately, the manual is easy to understand and use. For this reason, C programming novices and programmers with relatively simple window and data-entry needs will be fine with this package. In fact, since it is the easiest to understand and maintain, this may be the best package for those people. The inclusion of source code at the standard price makes C-scape worth considering.
—M.W.

Magus
Data & Windows

Data & Windows from Magus is supplied on two disks with a 230-page, IBM-sized manual. It is a new product (released in January 1987) and one of the few packages reviewed that includes support for interlanguage calling between Microsoft C, Pascal, and FORTRAN. While the list price is $345, discounts are generally available and an introductory discount of $100 has been extended indefinitely.

To define screens in Data & Windows, you use a stand-alone program to paint your screens and menus, defining fields and options for each field as you go. An excellent tutorial program covers building screens. When you save your screen to disk, Data & Windows writes a linker-ready file. You can also read a screen definition into an already running program if you don't want to link the screen into your application. This method is also

C and Pascal on:



We cut our teeth on UNIX, but have become famous on MS-DOS, which we enhanced with our UNIX-like **DOS Helper** ™ utilities: find (including tar), fgrep, cat, ls, mv, tail, uniq, and wc; and our superior optimizing compilers: **Professional Pascal** ™ and **High C** ™ on the PC are now well-respected by organizations such as Ansa, Ashton-Tate, AutoDesk, Boeing (BCS), Daisy Systems Corp., Deloitte Haskins & Sells, Digital Research, GE, IBM, Lifetree, Migent, Multimate, NYU, Silvar-Lisco, Sky Computers, Symantec, Xerox/Ventura, ...and *Computer Language* magazine; *Dr. Dobbs' Journal; PC Tech Journal; PC Magazine;* the *Journal of Pascal, Ada, and Modula-2*...

We supply the **first, and still only,** compilers generating **32-bit protected-mode code for the 80386** under MS-DOS. And our newly upgraded MS-DOS real-mode compilers were used by Symantec for their Q&A™ product to exploit the power of the 80386 real-mode instruction set. (**Just released:** HC v1.4 and PP v2.7, May 1987.)

Our **C Validation Suite** will blow your C compiler out of the sea, while our C compiler tracks the emerging ANSI Standard and generates tighter code with far better lint-like feedback help than competing compilers.

And you'll love **Professional Pascal**'s Ada-like packages, true data abstraction, C-like bit manipulation, and much more, along with the tight code that is linkable with **High C**, or other C, object modules (and vice versa).

Our **Translator Writing System (TWS)** goes far beyond LEX and YACC, with fully automatic error recovery...

All uniformly implemented on UNIX, VMS, CMS, MS-DOS, FlexOS...

Professional software developers in need of **industrial-strength tools** should contact:

MetaWare™ INCORPORATED

MetaWare Incorporated
903 Pacific Avenue, Suite 201
Santa Cruz, CA 95060-4429
(408) 429-6382 (META)
Telex: 493-0879 (META UI)

The Clear Choice for Large Programming Projects.

Name _____
Company _____
Address _____
City, ST _____ Zip _____
Phone: (____) _____

CL 05/87
Product:
Platform:

Circle what interests you:
PP HC TWS DOS Helper (DOS only
V.x 4.x DOS FlexOS VMS CMS
Sun Apollo Atari VAX 370
8086-family 80386pm 680x0 320xx

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others/owners: Ada/DoD; Apollo/Apollo; Atari/Atari; DEC,VAX,VMS/DEC; FlexOS,GEM-DOS/Digital Research Inc.; IBM,RT PC/IBM; MS-DOS/Microsoft; Q&A/Symantec; Sun/Sun; UNIX/AT&T.

Footnotes: 1. Atari, CMS versions available 7/87. 2. NS320xx version by special order. 3. UNIX not yet available on 370.

CIRCLE 64 ON READER SERVICE CARD

CL 06/87

required if you're using Microsoft FORTRAN v.3.3 because of language limitations.

A data-entry screen can be larger than the physical screen, up to 400 lines and 9,999 fields. Each screen has a window of from 1 to 25 rows and 1 to 80 columns. Fields can be up to 9,999 characters. The screen builder program includes a test mode that lets you run through a data-entry form without writing a program. You can also test drive your screen with only seven or eight lines of code.

To transfer data between the screen and memory, you can either move each field individually or define a structure that contains the variables in the same order and size as those defined on the screen. If you do this, only one function call is required to transfer your data between screen and memory structure.

Choices for field options are originally set using the screen builder program but can be overridden in the actual program using library calls. Menus, mouse support, display forms, and data entry forms are implemented. Optional (but handy) routines like a word-processing window, context-sensitive help, scrollable choice forms, and scrollable subfields are missing.

If you buy the package, have fun finding the phone number for Magus. The only places I found it were the sign-on screens of their programs and a small reference on page 57 of the manual. Once you do get through to technical support, you'll often be talking to the actual designer of the package. Presently, telephone support is unlimited. Text-editing windows are promised for v.2, with context-sensitive help available as a side effect. According to Magus, the upgrade fee to v.2 will be $25 to $75.

Data & Windows supports the following compilers: Microsoft, Lattice, Mark Williams, Aztec, DeSmet, and MetaWare. Only large memory models are supported. No royalties are charged to distribute application programs under DOS.

Data & Windows implements *string, date, char* (byte), *integer, word, long int, float,* and *double* fields. Boolean fields could be implemented easily using the byte-type field.

There is no picture for a particular field. Instead, you choose whether to allow alpha, numeric, alphanumeric, or any data type to be used. If you need more restrictive character processing, you can install a character string in a field validation routine that contains all valid characters for the field.

Up to 255 different strings can be created and installed; a string may be used for more than one field. Also, you can install a function pointer in a field that will verify the validity of a field, as well as provide for moving the field between the screen and your variable memory.

Data & Windows automatically includes a compiler's floating-point library. However, Magus supplies an object file that will satisfy the linker's floating-point references, so that the library is not loaded.

Data & Windows implements two types of menus: pull-down or standard. Selection from menus can be accomplished by first-letter entry or a cursor key-Enter combination.

Data & Windows can use a color (CGA or EGA) or monochrome monitor. Routines are provided to switch displays. You must pick color combinations that look right on both types of monitors, which can be done in the screen builder or during a program run. If done inside the program, you could install colors for fields and windows and override them if you are using a monochrome screen, for example.

If you're using an EGA or CGA that doesn't have a snow problem, an environment variable can be set to have the program update the screen as fast as possible. If you're running under Topview or Windows, a different environment setting causes the program to use the BIOS for screen calls.

I didn't encounter any problems using the package. I'd like to see more examples and information on the window library calls and accessing data. The manual does list examples of each particular function for C, Pascal, and FORTRAN. This package produced some of the smallest .EXE files of any of the packages, with a small address-list program ending up at under 30K.—M.W.

Greenleaf Software
Greenleaf Data Windows

Greenleaf Data Windows by Greenleaf Software is an extensive product. The version I received had five disks, a 225-page manual, and a reference card. The five disks consisted of one demo disk, two disks for Microsoft C v.3.0, and two disks for Microsoft C v.4.0.

All windows are logical and reside in memory. You can have up to 254 logical windows in the system at any one time provided you have the memory to do so. If you have created multiple windows, they can be displayed to the screen in any order. Windows are defined and created with the provided functions. If you want to create windows without writing the code yourself, Greenleaf Data Windows has *vbuild()* and *vedit()*.

The *vbuild()* function is an interactive window designer and window source code generator. Without actually knowing how to use the functions for creating windows, you can interactively design a window by invoking this utility function within a C program and produce the source code it would take to generate the window.

The *vedit()* function is an interactive text and data-entry form designer. With it, you can edit a specified window, working with text, prompts, data-entry fields, picture strings, and such. The editor creates two files: a window file with the window's contents and structure and a source code file that defines the fields. *vbuild* and *vedit* are well documented and can save you hours of coding.

Greenleaf Data Windows supports all memory models and formats of Microsoft C v.3.0 and 4.0. Also supported are Lattice C v.2.x, Lattice C v.3.x, Wizard, Computer Innovations, DeSmet, Aztec, and Mark Williams. Compiler support changes can be made for $75. No royalties are charged for application distribution.

Data types supported are *integer, unsigned, long, unsigned long, comma* (long integer), *double* (floating-point real), *string*, Boolean, *dollar, date,* and *time*. Greenleaf Data Windows supports pull-down menus, pop-up windows, and Lotus 1-2-3 style menus. All menu types can use an arrow key-Enter combination or first-letter match for item selection. Greenleaf Data Windows is compatible with Windows and Topview.

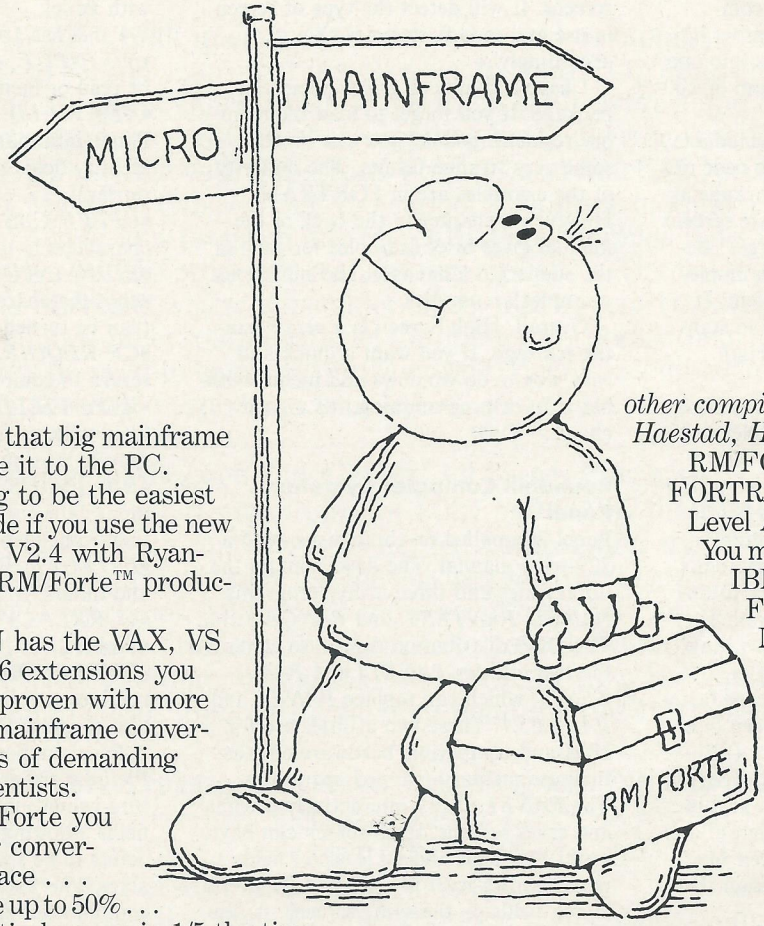
For pictures, Greenleaf Data Windows is patterned after dBASE III. The character "9" entered in a field allows only digits 0 through 9, "A" letters only, and so on. All other characters in a picture string are considered constants or fixed data.

Greenleaf Data Windows can automatically determine the type of monitor being used, with optional testing for monitor type also possible. It supports monochrome, CGA, EGA, Hercules, and Compaq dual-mode display.

I do have one minor gripe. The two utility functions *vbuild* and *vedit* are described on the manual's initial page, which referred me to the demo disk. The demo disk had no information on the two utilities; the only reference source on them was the function definitions in the manual.

Greenleaf Data Windows is a very detailed product. All of the functions have at least one example in the manual on how to implement and use each within a program. The supplied reference card is an aid in quickly learning about a function or attribute.—D.D.

FORTRANS-MIGRATION.



It's moving day.

Time to pack up that big mainframe program and move it to the PC.

Relax. It's going to be the easiest move you ever made if you use the new RM/FORTRAN™ V2.4 with Ryan-McFarland's new RM/Forte™ productivity tools.

RM/FORTRAN has the VAX, VS and FORTRAN-66 extensions you need, and is field-proven with more than two years of mainframe conversions by thousands of demanding engineers and scientists.

And, with RM/Forte you can complete your conversion at a record pace . . . cutting debug time up to 50% . . . diagnosing syntactical errors in 1/5 the time. You're in the fast lane because RM/Forte integrates all major program development tasks—edit, compile, link and debug—into one interactive, menu-driven environment. Just a few simple keystrokes move you from one task to another . . . and back again.

Plus, you get a full screen editor, a syntax checker and a source code manager that automates compiling and linking as you make changes.

RM/FORTRAN applications are more than just easy to develop—they run fast, too. Sieve or Whetstones, RM/FORTRAN benchmarks ahead of the bunch. And if you need it, RM/FORTRAN has a very fast 8087/80287 emulator to give you the power of a coprocessor.

But don't take our word for it. Listen to a moving professional: *"RM/FORTRAN lets us move mainframe programs to PCs faster and cleaner than any other compiler on the market." (John Haestad, Haestad Methods)*

RM/FORTRAN was the *first* PC FORTRAN GSA certified a Full Level ANSI 77 with no errors.

You may have heard of its sister, IBM PC Professional FORTRAN by Ryan-McFarland Corporation.

Enough reading. Start writing. Fill out and send in the coupon below to find out more about RM/FORTRAN with the new RM/Forte productivity tools.

Then move it!

Ryan-McFarland
609 Deep Valley Drive
Rolling Hills Estates, CA 90274
Or call 213/541-4828

RYAN-McFARLAND

CL-6

Let's get moving. Send me details on RM/FORTRAN V2.4 with RM/Forte Productivity Tools, and your special upgrade offer.

Name _____

Company _____ Title _____

Street _____

City _____ State _____ Zip _____

Phone ( ) _____

CIRCLE 65 ON READER SERVICE CARD

Softway
High Screen

The High Screen package comes in three parts: HS, DISPLAY, and LIBRARY. HS is a stand-alone screen editor for designing and creating screens and data-entry fields. With it, you can view the screens as they will appear in your application. The screens you design are saved as individual files that can then be used in application programs. Source code is not available or necessary.

The DISPLAY module is a memory-resident program that must be loaded prior to executing an application. DISPLAY intercepts standard output and looks for a specific set of escape sequences that signal DISPLAY commands. LIBRARY is a utility program for storing all generated screens into one large library file with a maximum of 200 screens per library.

High Screen is not a set of includable C functions, and it generates no code or object files. Softway bills the package as having "the ability to disassociate screens from your programming language." Because of the way High Screen is implemented, it is language independent. If the language can print or write to standard output, then you can use High Screen.

The review copy was supplied on five disks: two demo disks, an examples disk, the main program disk, and a utility disk. The utility disk contains two utility programs. *BEG* must be loaded before any other memory-resident program. *BEG* places a marker in memory that's used by the second utility, *END,* to unload the memory-resident program.

High Screen offers vertical scrolling and the ability to load a library of screens into memory (which means faster screen access) and choose between field-by-field or full-screen data entry (full-screen validates data once the full screen has been entered). High Screen also includes *FUNCTION,* a utility that allows the execution of any High Screen function in a batch file or directly from the DOS prompt.

High Screen supports several languages, including BASIC (both interpreted and compiled), C, Pascal, FORTRAN, COBOL, assembly, and dBASE II and III. The package implements the standard *integer, real,* and *date* data types. Also implemented are *alphabetic string, character string,* and *chosen character string.*

Alphabetic string only allows alphabetic characters, *character string* allows all characters, and *chosen character string* allows only characters specified in a user-created list with a maximum of 15 alphanumeric characters. Since High Screen is not connected to an application, the compiler's floating-point library is pulled in only if used.

High Screen allows you to create Lotus 1-2-3 style or pull-down menus. Within an application, a menu item can be selected using the arrow keys in combination with the Enter key or by pressing the first letter or number of a menu choice. High Screen can use monochrome, Hercules, CGA, and EGA screens. It will detect the type of screen in use and modify its color choices accordingly.

I have a couple of gripes about the package. If you forget to load the memory-resident module, you can expect some very strange results. The majority of the examples are in FORTRAN. However, a chapter in the back of the manual gives brief examples for each of the supported languages, and numerous examples are on disk.

Overall, High Screen is a very versatile package. If you want a quick and easy way to do windows and menus without a lot of programming, it's a good choice.—D.D.

Roundhill Computer Systems
Panel

Panel is supplied on three disks, with a 229-page manual. The disks contain the library files and three utility programs, *PANEL, PANTEST,* and *PANGENC.* The DOS distribution disks also contain the two utilities *PANELI* and *PANTESTI,* which can replace *PANEL* and *PANTEST.* These two utilities are for IBM and compatible hardware only as they use memory-mapped operations. The *PANEL* utility interactively designs and saves screens. Each screen can have up to 255 fields and 10 levels of fields per screen (a level is a method for overlaying fields on the same screen). A separate file contains information about visual attributes, screen functions, screen size, keyboard functions, and such. If you use the *PANELI* memory-mapped version, this external file is unnecessary.

PANTEST tests and exercises all generated screens and allows you to test all input fields and validations. All screen testing and debugging can be done with the *PANTEST* utility.

The *PANGENC* utility generates the C code that describes the screen created. Normally two files are created: a C source code file and a header file. The source code file defines all the structures and data areas of the screen, while the header file is included in an application to give access to the field data areas.

Panel supports the following compilers: Microsoft v.3.0 and v.4.0, Computer Innovations C86, DeSmet, Mark Williams C86, Wizard, and Aztec. Also supported are Zilog Zeus, XENIX 286, and generic UNIX. All memory models are supported.

The libraries were compiled for use on an 8086; 80286 libraries are available at an additional charge. Panel is also available for Microsoft FORTRAN under PC/MS-DOS, Microsoft BASIC or BASICA under PC/MS-DOS v.2.0 and later, and the Commodore Amiga. Other interfaces are available for v.2.x of Lattice or Microsoft C. Applications can be distributed without royalties, but you can't distribute any source code provided with Panel.

Panel has 10 field-entry attributes:

- *NO INPUT.* A fixed field that will not be read or included in any data record.
- *KEY FIELD.* An attribute used by the Panel data management program to identify fields to be used for data retrieval.
- *UPPER CASE.* Converts all lowercase characters to uppercase.
- *ECHO INPUT.* A default attribute that echos the characters as they are typed (can be turned off).
- *CR REQUIRED.* Requires a carriage return to complete the field.
- *WIPE FIELD.* Causes a field to be filled with its fill character after entry to prevent editing of the field.
- *RIGHT JUSTIFY.* Does not affect user input data until the field is completed, then right-justifies data.
- *INTEGER.* By default, 0 through 9 and minus.
- *CURRENCY.* Two decimal-place numbers.
- *UNDEFINED.* Available for validation routines; scrollable selection fields are also supported.

Panel supports vertical pop-up menus. Pictures are available through the picture-validation extended attribute for fields, allowing different validation characters to be specified for each field character. The *TAILOR* utility creates a configuration file, used by *PANEL, PANTEST,* and your application, that contains the screen and keyboard definitions.

As for problems, the manual is somewhat difficult to follow. However, the demo program on the distribution disks is a very good example of how to implement different functions.

Late in the review process, I also received a beta test copy of Roundhill's Panel Plus v.0.92, an enhanced version of Panel. Panel Plus's library functions are smaller and faster than the equivalent functions in Panel. All Panel screen layouts are upwardly compatible, and full

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

Pascal-2™ FOR MS-DOS

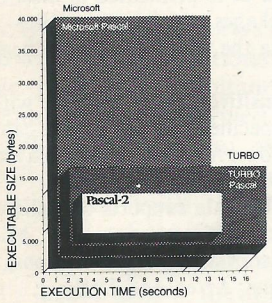
Have you reached the limit with TURBO?

List: $395
Ours: $359

Upgrade to Pascal-2.

- Cut execution time by 200% over TURBO
- Cut executable program size by up to 50%
- Transport MS-DOS programs to VAX, PDP-11 and 68000 machines with only minor adjustments, and vice versa
- Speed error corrections and save development turn-around time with sophisticated error checking and reporting
- Use all of DOS-addressable memory through efficient large-memory model
- Access DOS services and network files
- Call Microsoft FORTRAN, C, Pascal and assembly routines

Quicksort Benchmark



UPGRADE WITHOUT LOSS – easy migration path from TURBO with compatible strings, equivalent procedures & access to TURBO graphics

Plus **Pascal-2 gives you more for the dollar! You get these extra features included free:**

- Interactive source-level debugger
- Error walkback to trace errors to their source
- Intel CEL87 math library
- High-level profiler

The following are trademarks: Oregon Software, Pascal-2, Oregon Software, Inc.; Intel, CEL87, Intel Corporation; MS, Microsoft Corp.; TURBO Pascal, Borland International, Inc.; PDP, VAX, Digital Equipment Corp.; MC68000, Motorola, Inc.

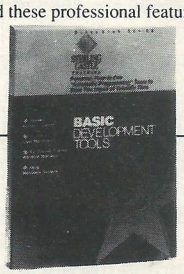
OREGON SOFTWARE

NEW! From Sterling Castle...
BASIC Development Tools™

Powerful "Automatic Programming" Tools That Save You Hours of Valuable Time.

The novice or power programmer, can easily add these professional features.

- **Screen Builder**
- **B+ Tree**
- **EZ Screen Pop-up Windows**
- **Help Message System**



BASIC Development Tools™ (BDT™) is compatible with the newest, fastest compilers, including Microsoft QuickBASIC™ and Borland Turbo Basic.™ In BDT you have four powerful aids that can be used separately or together.

Screen Builder System translates the painted screen image into BASIC code which then can be merged in your program.

B+Tree Data Manager is a very fast data file index system providing both direct and sequential access to data. Complete source provided.

EZ Screen Pop-up Window Manager, written in assembler, easily inserts menus, windows, notepads. Saves a portion of the screen to/from a buffer.

Help Message System allows the creation of context sensitive help messages in your application program.

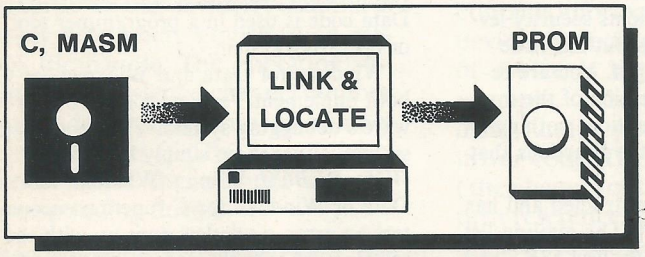
With BDT you have four of the most popular programming aids for **$99.00.** Includes two diskettes and 220 page manual.

60 DAY FREE TRIAL.
ORDER TODAY! (800) 722-7853
(213) 306-3020 in California

STERLING CASTLE™
Sterling Castle, 702 Washington St., Suite 174, Marina del Rey, CA 90292

All trademarks acknowledged.

FIRMWARE DEVELOPMENT



LINK & LOCATE enables PC users to produce ROM-based firmware for 8086/87/186 from object files generated by popular C compilers, such as from Wizard, Microsoft and Lattice, and MASM assembler from Microsoft. Provides full control of segment placement anywhere in memory. Supports output of Intel HEX file for PROM programmers, Intel OMF absolute object file for symbolic debuggers and in-circuit emulators. Includes Intel compatible linker, locator, librarian, hex formatters and cross reference generator. $350.

SYSTEMS & SOFTWARE INC.

Our: $329

3303 Harbor Blvd., C11, Costa Mesa, CA 92626
Phone (714) 241-8650 FAX (714) 241-0377 TWX 910-695-0125

COBOL PROGRAMMING PRODUCTIVITY!

screenplay™

SCREEN MANAGEMENT SYSTEM

SUBSTANTIALLY REDUCE APPLICATION DEVELOPMENT TIME!
Save valuable time by avoiding the tedious, time consuming process of writing screen handling source code. *screenplay's* easy-to-use panel painter allows you to create I/O panels, pop-up help windows or menu panels which you use in your program.

FLEXIBILITY IN PANEL PAINTING; FLEXIBILITY AT RUNTIME!
True screen handling flexibility is yours with *screenplay*, because you can override default panel settings to design practically any type of panel imaginable! *screenplay* continues to provide the flexibility you need during the operation of your program by allowing you to change just about any panel characteristic at runtime.

PROTOTYPE PANELS BEFORE YOU WRITE CODE!
With *screenplay*, you can prototype your draft screens before you write a single line of COBOL source code. These can then be reviewed with your boss or your customer for final approval, before you start writing source code.

MORE THAN ONE LINKING OPTION!
In addition, linking *screenplay's* runtime unit is your choice! You can link *screenplay* by interrupt or directly to your application. And if your compiler doesn't allow a direct link, you can take advantage of a dynamic load option for linking *screenplay* to your application.

FULL CONTROL OVER KEY ASSIGNMENT!
You can assign practically any keyboard key to serve as a specific cursor function and define exactly which keys will return control to your application. *screenplay* gives you the power to entirely reconfigure the keyboard for your program.

POWERFUL, ONE-STEP PANEL PAINTING
screenplay's panel painting process is a "one-step" approach. There's no need to go through a separate process to establish fields on your I/O panel. What's more, you can use any ASCII character in your *screenplay* panels. You also have full control over character attributes such as foreground and background color, intensity and blinking.

EASY PANEL FILE MANAGEMENT!
Panels are stored in an ASCII file which is compressed to save memory and disk space. *screenplay's* Panel Management Facility allows you to easily copy panels across and within files, rename panels, delete panels, test and print panel details. You can even print an image of the panel for your documentation!

NO ROYALTIES / NOT COPY PROTECTED!
screenplay's panel control runtime unit, better known as The Panel Control Facility may be linked to your program without paying a dime in royalties. In addition, *screenplay* isn't copy protected to make it even easier-to-use. The Panel Control Facility allows you to control almost every panel characteristic by using parameters.

SUPPORTS MANY COBOL COMPILERS!
Supports IBM COBOL, Microsoft COBOL, Realia COBOL, Ryan-McFarland COBOL and Ryan-McFarland COBOL 8X; List Price $175.00, **OUR PRICE $155.00**

Flexus — *PROFESSIONAL TOOLS FOR PROFESSIONAL PROGRAMMERS*

Call Today for FREE detailed information or try Risk-Free for 31 days.

800-421-8006
HOURS: 8:30 A.M. - 8:00 P.M. E.S.T.

THE PROGRAMMER'S SHOP
Your complete source for software services and answers
5-L Pond Park Road, Hingham, MA 02043
Mass: 800-442-8070 or 617-740-2510 2/87

CIRCLE 66 ON READER SERVICE CARD

library source code is included with the new version. Panel Plus's manual is much easier to read and understand. —D.D.

Creative Programming Consultants Vitamin C

Vitamin C is supplied on two disks and one manual with 380 IBM-sized pages. Screens are defined by standard function calls to the Vitamin C library. The company has a second product, VCScreen, for $99 that allows you to paint your screens.

Screens are defined by opening a window, then calling the *SAY* and *GET* functions to define the text and fields on the screen. Finally, the *READGETS* function call will process your *GET* list.

Vitamin C has many different levels of function calls, each requiring a little more information than the previous one. For example, the highest level always deals with the current window and uses the current display attributes, the next level might specify different display attributes, and the lowest further specifies the window. You can mix different levels in the same window.

Vitamin C implements scrollable data-entry fields and word processing windows. The system has plenty of hooks for you to install your own validation functions. The read-a-key function in Vitamin C is written to allow some background processing while waiting for a key to be pressed. This would allow you to keep a clock on the screen, for example. You can also install your own key-processing function that can look at each key and process it as it sees fit.

Vitamin C has a context-sensitive help function. Your help text is placed in a text file with keywords marked off by @@. When the user presses the help key (which is redefinable) in a field, the text for the help keyword attached to this field will be displayed in a help window. A default keyword can be set for those fields where no help keyword is available. Each field can also contain a status message that is displayed in a window whenever that field is active.

Support for Vitamin C is very good, and a bulletin board system has been set up for technical support. Anyone can call to get examples and information on Vitamin C. When you register your product, your security level will be increased to allow you full access to all areas of the board for code fixes and more specific examples of various functions.

Vitamin C supports the following compilers in all memory models: Microsoft, Lattice, Aztec, and Computer Innovations. No royalties are charged to distribute your application programs under DOS. UNIX and XENIX versions are promised for the summer of 1987.

Vitamin C implements *string, integer,* and *double* fields. Valid picture characters will allow any character, alphanumeric characters, or numbers. The any-character and alphanumeric options can also specify if lowercase should be translated to uppercase.

Additional field types would be nice but are not absolutely necessary. You can specify date fields with a picture of *99/99/99* and a validation function of *valdate()*. The compiler's floating-point library is pulled in whether or not you use *double* fields. Since you get source code, you can modify the routines, but it takes work.

Vitamin C implements two types of menus. You can choose one-line menus with pull-down submenus or standard menus. Selection from menus can be accomplished by first-letter entry or a cursor key-Enter combination. In addition, the programmer can specify single keys to call a function and bypass the menus.

Vitamin C can use either a color (CGA or EGA) or monochrome monitor. When the window system is initialized, the machine configuration is checked for the monitor card type. If you're using a monochrome monitor with a CGA card, a function is provided to override the color choices and use colors appropriate for monochrome monitors instead.

The package implements security levels for menus and fields but has little documentation on using it. You are referred to the tutorial section of the manual, but the tutorial mentions nothing on security levels or even the functions that use them.

Vitamin C is well constructed and has an easy-to-read manual. The mailing-list program discussed earlier was 53K using the small memory model version, the only package capable of doing it in small model without complaints from the linker about segment size being exceeded. —M.W.

Vermont Creative Software Windows for Data

This system is supplied as two packages: Windows for C and Windows for Data. Windows for C is a stand-alone package that implements a windowing environment in C that's compatible with Topview and Windows. Other features include mouse support and text-file editing.

Because Windows for Data uses Windows for C for its window management, all of the features offered by Windows for C are available in Windows for Data but usually with greater functionality. Source code is available but very expensive.

Fortunately, few people will need the source code. Most parameters can be changed by modifying the *#include* files. The documentation details such things as adding validation functions to each field, creating a new field type, and changing picture characters and error messages.

Windows for Data is definitely a heavyweight package. I didn't find a function in any of the other packages that wasn't implemented in this package. In addition, Windows for Data has a few additional features like scrollable subforms (useful for invoice line items) and scrollable field-entry selection (where a list of allowed responses to a particular field can be put in a window and selected with arrow keys). While both of these features could be implemented in most of the other packages with work, in Windows for Data the work has already been done for you.

Context-sensitive help is available with a developer-specified key. Each field can also have its own help message displayed whenever the cursor enters a field.

Windows for Data supports the following compilers in all memory models: Microsoft, Lattice, Computer Innovations, Mark Williams, Aztec, and DeSmet. Versions of Windows for Data are also available for UNIX, XENIX, and other terminal-based operating systems. No royalties are charged to distribute application programs under DOS, but distribution fees are charged for UNIX and XENIX applications or if Windows for Data code is used in a programmer tool or operating system.

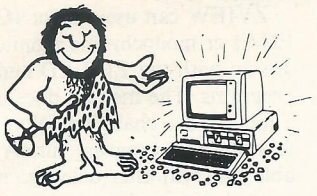
Windows for Data and Windows for C both implement Vermont Creative Software's debugging system. The debugging system is turned on simply by defining *WN—DEBUG*. When a Windows for Data or Windows for C function encounters an error, a window pops up with the sequence of calls that generated the error along with an explanatory error message. You can easily incorporate this error system into your own functions. When the program is fully debugged, the debugging system can be removed, resulting in a space savings of about 11K.

Windows for Data implements *string, date, time, char,* Boolean, *integer, long int, menu, float* (*double*), and *decimal* fields. Menu fields are used in menus only. Support for unused field types is easily removed, thus avoiding needlessly pulling in floating-point libraries.

Windows for Data implements many different types of menus: Lotus 1-2-3 style, pull-down, or standard. A menu can call other menus or user-supplied functions. Overall, the menu processing

Professional Programming Products

by BCSoft



FREE

PC-WRITE™ text editor
with every purchase.

PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS!

NEW

Quick-Tools™
for QuickBASIC™ users!

- A comprehensive library of over 90 subroutines and functions directly CALLable from your QuickBASIC programs.
- SORT routines allow sorting of single and multidimensional arrays.
- Binary search functions.
- Unique screen handling functions allow splitting screens, fast scrolling, special string printing, character attribute control, phantom cursors, etc.
- Keyboard scanning and status calls, with field inputing and decoding. On screen editing of input fields.
- Unique file handling routines.
- FREE Updates and phone support!
- Plus much, much more!

Only $129.95 Complete

NEW

NET-TOOLS™
NETBIOS Programming Tools

- NET-TOOLS allows you to write programs for ANY NETBIOS compatible local area network - fast and easily. CALLable from Microsoft Assembler, C, PASCAL, or FORTRAN.
- Add and Delete local names.
- Initiate and Cancel sessions. Just give NET-TOOLS the name of the computer you would like to call, and it will automatically locate the user and make the connection for you.
- Transfer Messages with automatic retries and error detection. Both the datagram and session protocols.
- Redirect Local Devices simply and easily with a single function call.
- Plus much MORE !
- Complete SOURCE CODE is provided, FREE !
- FREE Updates and phone support!

ONLY $149.00 Complete

TURBO.ASM™
For Turbo PASCAL users !

- A unique programming tool which is a must for every Turbo Pascal user. The only package designed for interfacing assembly language with Turbo Pascal.
- Outlines several different ways to add assembly language routines to Turbo programs, some without effecting your code space.
- Fully explains the internal workingsof Turbo Pascal and data passing methods.
- Includes a library of Assembly routines which can be used directly from Turbo Pascal.
- The best way to learn assembly language.
- FREE Updates and phone support!

Only $99.95 Complete

ASMLIB™
The Programmer's Library

- A set of over 210 subroutines to greatly increaseyour productivity. Written in assembly language anddirectly CALLable from Microsoft Assembler, C,PASCAL, and FORTRAN.
- Complete SOURCE CODE provided -FREE!
- Text WINDOWing functions allow up to 64 overlapping windows.
- TERMINATE and STAY RESIDENT programs arewritten easily. Programs can "POP UP" with a simplekeystroke. Use DOS calls from them, also.
- GRAPHICS on the EGA, CGA, and Hercules™ Monochrome.
- Virtual file functions.
- Full FLOATING POINT math and trig with 8087 support.
- Int. driven async. support, plus MUCH MORE!

Only $149.00 Complete.

NO ROYALTIES REQUIRED

To ORDER call or write to:

BC Associates
A division of BCSoft Corporation
3261 N. Harbor Blvd.
Suite B
Fullerton, CA. 92635

VISA, M/C, or COD orders are welcome!

CALL TOLL FREE 1-800-262-8010 in CA. dial 1-714-526-5151

CIRCLE 67 ON READER SERVICE CARD

is extremely good. Windows for Data has predefined picture types for letters and spaces, printable characters (including international language characters), and numbers in scientific and decimal formats.

Windows for C automatically senses the monitor being used and sets itself to efficiently display on that monitor. If a CGA monitor is being used, the program will wait for the vertical retrace signal before writing to the screen to eliminate snow. Colors for fields, forms, menus, and windows are defined using logical attributes. This allows the Windows for Data library to display in color on a color monitor while still making intelligent choices when displaying on a monochrome monitor.

Windows for Data also checks whether a program is running under Topview or Windows. If it is, the package uses the proper screen updating procedures so that your program can run in a window or in the background. If not, it will write directly to the screen memory for faster updates. The Topview and Windows check can be disabled if desired.

My only problem with Windows for Data is the manuals. The function listings are not written in standard C style. Function names should be on the top outside corner of the page so you can more easily find the function you want. The function parameters are not listed at the top of the page. Instead, you must look further down the page or even a few pages away for the parameter listing.

Many of the function calls use defined constants for field and border types, colors, and such, but the allowable constants are usually not listed with the function. You must switch to a different chapter or even a different manual to find them. This searching can be a big chore since there are two manuals (one for Windows for C and one for Windows for Data), each with more than 300 pages. More information on return values is needed.

Finally, a brief example showing the usage of each function would be helpful. .EXE file size for the small address-list program was about 75K under Microsoft C v.4, medium model.—M.W.

Data Management Consultants ZVIEW

ZVIEW is supplied on two disks with an 80-page, IBM-sized manual. Screens are defined through a two-step process. First, you use ZPAINT to paint your screens and define text and field placement, then ZFIELD is run to customize different fields. Your screen is saved in a special format file and a header file. This two-step process is cumbersome; other packages have an external program to define the screen and fields at the same time.

The two screen definition programs are definitely not for the faint at heart. ZPAINT's documentation has lots of warnings about using test screens until you've figured out how to use a command line parameter that allows saving field characteristics. You must specify whether to link field characteristics to fields by field positions or field numbers. If you do not capitalize the command line parameter, the program aborts. It's not hard to force the command line to caps—it only takes one line of code. Still, other programs with screen painters do not have this restriction; data fields can be moved around on the screen at will, with all field attributes remembered.

ZPAINT can only handle one screen at a time. If you need more than one screen to hold your data or help screen, you save each screen to a file and then use the DOS *COPY* command with the */B* option for binary concatenation to combine them into one file. Other programs provide for a screen larger than the display by scrolling the window over the virtual screen.

To define your fields, you choose one of three field types: unprotected, protected, or heading. Then, depending on the field type desired, you choose a function key to make the field.

To change the colors of fields, keep the manual handy. Function keys are used to define the foreground and background colors. The program has very little error checking and bombs easily. The keys chosen by ZPAINT to move about the screen are sometimes not readily apparent, although you can override most of them in your own programs. Also, many topics in the manual are unclear. The ZFIELD program is similarly cryptic.

ZVIEW supports the following compilers in all memory models: Microsoft, Lattice, Aztec, and Datalight. The package implements *character, byte, int, long, float, double, string,* and *decimal* types. The *decimal* type uses a *long int* to store a *float* number with an implied decimal point.

ZVIEW has no picture strings in the normal sense. Instead, ZVIEW has *nonprotected, protected,* and *heading* types. *Heading* and *protected* are in essence the same but are split up so they can be dealt with separately. Support is provided for scrollable list entry.

Floating-point library routines are pulled in whether you use them or not. Menus are not supported in the manual at all. They would be difficult if not impossible to implement with the supplied library functions.

ZVIEW can use a color (CGA or EGA) or monochrome monitor. Colors are set with the ZFIELD and ZPAINT programs. The manual does not provide a method for changing colors or screen text during a program run. A global variable that will alter colors to monochrome equivalents can be set at run time.

I can't recommend ZVIEW due to its limitations. Defining your screens is a complicated two-step process. The programs are not well documented and depend on files being in specific places on your disk. The inability to support menus and redefine screen text and colors at run time is a big mark against the package.—M.W.

The big picture

Most of these packages could be recommended for specific applications. If you do a lot of interlanguage programming with Microsoft languages, Magus's Data & Windows is a good choice. If your windowing needs are simple and C is your only language, Oakland Group's C-scape should be considered.

If you routinely write in many different languages, you should look at Softway's High Screen. If you require full graphics support, Sydetech's C-Display Manager is worthy of mention. However, if text graphics is enough for your application, consider a different package. C-Display Manager's I/O services are not well thought-out and don't include much in the way of validation routines. Most of the work on this package seemed to go into the graphics area.

For full-featured packages, only one package can be easily recommended: Vermont Creative Software's Windows for Data. This package includes everything you need, except an easy-to-use manual. Unfortunately, this package also requires the most work to use. If you need source code for your application, make sure your wallet is wide open or get Vitamin C. As stated, ZVIEW can't be recommended. The other packages have more functionality, are easier to use, and cost less.

Picking the best value package is hard; Creative Programming Consultants' Vitamin C and Greenleaf's Greenleaf Data Windows are very close. If you're a source code fanatic like me, Vitamin C is preferable. It gives you complete source code for $225, while Greenleaf only supplies object code for the same money. These two packages were the reviewer's personal favorites. Panel Plus takes third

place based on features and price. We highly recommend you get Panel Plus over Panel because of the manual changes and enhancements to the library functions.

Many packages do a reasonable job of implementing data-entry windows. We looked for the most bang for the buck. If money were no object, our selections would be different. It all depends on what features you need, your price range, whether source code is necessary, the support a company offers, and your present and future choices of languages, compilers, and operating systems.
—M.W.

Donald Drumheller is director of computer services for City University, Bellevue, Wash. Drumheller is an applications programmer who works in BASIC and C.

Michael Wilson is president of Wilsoft Inc. in Tampa, Fla., where he works on bar-code programming tools using C and assembly.

C screen managers manufacturers

C-Display Manager—$179
C-Display Librarian—$99
Display Express—$99
Sydetech System Development Technologies Inc.
43-23 Colden St., #17C
Flushing, N.Y. 11355
(718) 886-0221

C-scape
Package—$199
Package with Dan Bricklin's Demo Program—$259
Oakland Group Inc.
675 Massachusetts Ave.
Cambridge, Mass. 02139-3309
(800) 233-3733

Data & Windows
Package—$245
Package with source—$595
Magus Inc.
10027 Spice Ln., #1703
Houston, Texas 77072
(713) 498-2856

Greenleaf Data Windows
Greenleaf Software
1411 LeMay Dr., Ste. 101
Carrollton, Texas 75007
(800) 523-9830
(214) 446-8641 in Texas and Ark.

High Screen—$129
Softway Inc.
500 Sutter St., Ste. 222
San Francisco, Calif. 94102
(415) 397-4666

Panel
Package—$295
Source code—$500
Panel Plus—$495
Roundhill Computer Systems
American distributor:
Lifeboat
55 S. Broadway
Tarrytown, N.Y. 10591-9990
(800) 847-7078
(914) 332-1875 in N.Y.

Vitamin C—$225
Creative Programming Consultants
Box 112097
Carrollton, Texas 75011-2097
(214) 416-6447, (214) 245-6090

Windows for Data
Package—$395
Source—$1,290
Vermont Creative Software
21 Elm Ave.
Richford, Vt. 05476
(802) 848-7738, (802) 848-7731

ZVIEW—$245
Data Management Consultants
5325 S. Valley View Blvd., #7
Las Vegas, Nev. 89118
(800) 423-0930

JACK™ The Resident Program Developer's Kit.

JACK™ contains everything you need to create your own RAM resident software without the headaches of memory management, windows or DOS re-entrancy.

JACK™ FEATURES:

Virtually any C or ASSEMBLER program can be made memory resident without any modification to the existing source code. All you have to do is link your program with our library.

Screens are automatically saved to conventional memory [or the expanded memory of an Intel Above Board if present] when popping up and restored upon popping down .

DOS function calls and disk I/O within your memory resident program can safely be used and JACK™ does not make any use of undocumented DOS calls to solve the DOS re-entrancy problem.

All programs developed with JACK™ peacefully coexist amongst themselves and with the many popular pop-up utilities from other developers.

CrackerJack would like to introduce BEYOND™, a library of memory management and service routines for the Intel Above Board. Now all of your programs can take advantage of expanded memory. With our library and Intel's ABOVE Board, your programs can go ABOVE and BEYOND™ their normal limitations.

To Order, please send $199.95 [US] + S/H for JACK™ or $49.95 [US] + S/H for BEYOND™ by certified check or international money order to:

Crackerjack Microsoftware Corporation
200 Bay Street PO Box 86
Toronto, Ontario M5J-2J2
Canada

(416) 865-9621.

VISA

Please Note: When ordering, please specify either the Assembler version or C version of JACK™ and indicate the vendor and release level of your compiler.

JACK™, CRACKERJACK and BEYOND are registered trademarks of CrackerJack Microsoftware Corporation, Intel is a registered trademark of Intel Corporation, Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.

CIRCLE 69 ON READER SERVICE CARD

C CODE FOR THE PC

source code, of course

C Source Code

FSP (screen manager) $400
GraphiC 3.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy) $300
Essential C Utility Library (400 useful C functions) $160
Essential Communications Library (C functions for RS-232-based communication systems) $160
Panache C Program Generator (screen-based database management programs) $125
B-Tree Library & ISAM Driver (file system utilities by Softfocus) $100
The Profiler (program execution profile tool) $100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library) . . $90
ME (programmer's editor with C-like macro language) $75
Wendin PCNX Operating System Shell $75
Wendin PCVMS Operating System Shell $75
Wendin Operating System Construction Kit $75
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.) $70
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.) $70
EZ_ASM (assembly language macros bridging C and MASM) $60
Make (macros, all languages, built-in rules) $50
Coder's Prolog (inference engine for use with C programs) $45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages) $45
Biggerstaff's System Tools (multi-tasking window manager kit) $40
TELE Kernel (Ken Berry's multi-tasking kernel) $30
TELE Windows (Ken Berry's window package) $30
Clisp (Lisp interpreter with extensive internals documentation) $30
Translate Rules to C (YACC-like function generator for rule-based systems) $30
6-Pack of Editors (six public domain editors for use, study & hacking) $30
ICON (string and list processing language, Version 6 and update) $25
LEX (lexical analyzer generator) $25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor) $25
C Compiler Torture Test (checks a C compiler against K & R) $20
A68 (68000 cross-assembler) $20
Small-C (C subset compiler for 8080 and 8088) $20
tiny-c (C subsubset interpreter including the tiny-c shell) $20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp) $20
List-Pac (C functions for lists, stacks, and queues) $20
XLT Macro Processor (general purpose text translator) $15
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner) $15

Data

Webster's Second Dictionary (234,932 words) $60
U. S. Atlas (29,000 cities with retrieval program) $40
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format) $30
The World Digitized (100,000 longitude/latitude points) $30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts) $15
U. S. Map (15,701 points) $15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders MasterCard/VISA

CIRCLE 68 ON READER SERVICE CARD

Turbo Pascal prototyping and screen management

By Jim Powell

New computer users have a hard time describing just exactly what it is they want from a new system. Even experienced users find visual aids helpful in describing their needs. If users want to see what a system will look like before they decide what they want (or don't want), it may be time for a prototyping tool.

This wrap-up covers four Turbo Pascal prototypers with an eye to their simplicity, speed, and flexibility. I also judged them on turnaround time from screen design change to compiled prototype. The products covered are The Research Group's Saywhat?!, The Software Bottling Co. of New York's Screen Sculptor, Sophisticated Software's TurboMAGIC, and Hawaiian Village Software's Turbo Master.

As prototypers, these products let you and your users see screen layouts for data entry and display. They allow mixing text fields, input and display fields, graphics, and moving the cursor from field to field to simulate data entry or an inquiry.

All four create screens that can be accessed using Turbo Pascal code. All are designed for painting screens with text or input fields; foreground and background colors are selectable for fields or entire screens.

All produce Turbo Pascal source for testing, though the code may need modifications to run. You may want to use generated code as the basis of your application program, adding bells and whistles (such as extending the validation criteria) as desired. The test program allows you to run a prototype, giving you and your users an idea of how the cursor moves and the types of error, help, and prompt messages available for each field.

Each Turbo Pascal prototyper provides screens as picture files to be read at run time. Saywhat?! and TurboMAGIC also allow you to create code for screen layouts that can be incorporated into your source code. This feature takes up space but speeds execution. While all can modify generated source code, this lengthens turnaround time.

All work in fundamentally the same way. When invoked, menus or commands lead you to a blank screen with limited displayed information (for example, cursor position, current color selections, and file name). By using function keys, menus, or commands, you invoke graphics characters, insert or delete fields or rows, and specify field attributes (such as Turbo Pascal field name and type, edit masks, and prompt messages).

Saywhat?! generates procedural code when a screen is saved; the others require a further step using a code generation function. You specify which screen files to use and the program writes the code. Code can be skeleton code you incorporate into your program or complete, ready-to-compile source.

Screen Sculptor points out actual and possible errors as it encounters them (for example, two different screens with the same field name). Invoke the Turbo Pascal compiler on a source file and you have a prototype the user can examine and play with until the screen looks just right. In most cases, the features generated can be changed. For the wrap-up, I used generated code with the products' default values.

You should keep in mind that users may expect a finished product to look just like the prototype. If they do, products should generate finished code.

Criteria for selecting final software development tools can be very different than those for prototyping tools. Once users sees a prototype, that vision of how things work can be stuck in their mind. When choosing a prototyping tool, you may want to ask how useful the product is for developing a finished application.

All the products have easily invoked help screens and can create and edit screens. All run on an IBM PC/XT/AT or compatible with DOS 2.0 or later. All can run on a two-floppy system, but this is practical only for Saywhat?! and Screen Sculptor. None of the products is copy protected or requires royalties for created programs.

I should define some terms before describing the products in detail. First, walking order is the order in which the cursor visits input fields on the screen. All the products allow you to change their walking order so that the cursor can go down a column of input rather than across a row. The default walking order is usually the same order used when placing fields on the screen.

Painting is the ability to select a background color, position the cursor where you want this new color to appear, and use the arrow keys to select the area for the color change. When in paint mode, you can paint an entire rectangular block or paint a one-row-by-one-column area (called a square) at a time.

Graphics characters are all the extended graphics characters, like happy faces, diamonds, and the lines used for boxes.

Finally, a text field is anything that's not a user input or display field. For example, *CUSTOMER NAME:* is a text field, but the space for entering or displaying a name is an input or display field. Display fields are defined (that is, space is reserved and attributes are given) but skipped when the cursor visits a field.

For purposes of comparison, I used each product to create a simple screen with a double-line box surrounding the 24-row screen image. My test screen had room for a 20-character last name, 15-character first name, social security number, hire date, annual salary amount (lower bound of $5,000 and upper bound of $100,000) with two decimals, and sex code (male/female).

All the test screens used identical text fields ("Enter last name:" for example). However, TurboMAGIC insisted on a six-character value for the male/female input field. The purpose of the test screen was to generate the basic code to step the cursor from one field to the next and recognize bad input. Only system-generated error messages were used; I didn't incorporate help screens. Each product's default settings were used in all cases.

All programs were compiled using Borland International's Turbo Pascal v.3.01A on a Zenith 158 under MS-DOS 2.11. All screen images were kept in a

separate PICture file and read by a program whenever a screen was displayed. The numbers for total lines compiled and size of screen files in Table 1 on p. 139 are for this simple program.

The Research Group Saywhat?!

Saywhat?! from The Research Group allows you to create screens with some time-saving and sophisticated tricks. It's not menu driven—you'll need to keep the reference card handy—but the command mnemonics are well thought-out.

The Saywhat?! editor is quick and easy to learn. You can copy, move, insert, and delete lines, blocks, and fields within blocks. Creating boxes is a matter of choosing single or double lines, then marking the upper and lower corners of the box. Corners are handled automatically.

Saywhat?! supports scrollable boxes, saving positioning time, and rubber-stamping blocks, which can be used for creating columns of fields with identical attributes. Graphics characters can be selected by ASCII number or from a menu. The editor's only quirk is the use of column 0 instead of the conventional column 1 for the leftmost column on a screen.

Saywhat?!'s terminology is heavily influenced by dBASE. The *GET* and *SAY* commands define input and output fields. These commands put a propeller symbol on the screen at the first position, make a guess at the field name based on the text field, and allow you to describe the edit mask (which also determines the field length).

While the dBASE orientation should be no problem for most programmers, the documentation has a number of lapses. For example, edit masks are not mentioned in the manual nor are there sample files on disk. Programmers unfamiliar with dBASE will have no clue as to what these masks are. The edit masks did appear in my generated test code and caused a Turbo Pascal compile error.

Saywhat?!'s help windows are actually subscreens. The original screen is saved to a page area and can be restored when the help window disappears.

Choosing colors is easy. The paint command can work on a whole screen or within a box. The current background/-foreground combination is displayed at the top of the screen.

A screen can be written to a text file, which can be incorporated into documentation (assuming your word processor will support any special graphics characters used). You can also display the screen directly from DOS by typing *POP <Screen file name>*.

A DOS command allows jumping in and out of Saywhat?! Thus you can jump to the Turbo Pascal compiler, test your results, and jump back in with minimal effort. The nicest feature is the undo button, which acts as expected; you can even undo the undo.

Saywhat?! supports BASIC and dBASE II and III (including III+). Minor modifications can be made to use the Clipper compiler, and instructions are included for Microsoft's C compiler. A batch file can be used to display screens from the DOS prompt so no compiler is actually necessary for simple screen displays like slide shows.

IMPORT is a memory-resident program that captures a screen image and converts it to Saywhat?!'s format. If you have a program with screens similar to what you want, you can invoke IMPORT, tweak the screens, and quickly show users the results.

The memory-resident program VIDPOP is loaded before you run your prototype. To call up VIDPOP to display a screen, you use two ASCII 255s and the name of your screen file. Screen display is quick, without snow, and adjusts to different monitor types (color, monochrome, EGA, and Hercules and Paradise graphics cards). Documentation on the Saywhat?! disk describes how to incorporate VIDPOP into your Turbo source (a feature not tested).

Unfortunately, Saywhat?! has a number of serious problems as a prototyper. When a field definition is complete, all that remains on screen is the propeller at the first field position—there's no visual clue to the length of the field.

Partly because of this poor design, fields can be overlaid on other fields. My *CUSTOMER NAME* field was 20 characters long; without any visual clues to its length, I began the *ZIP CODE* definition right in the middle of it. No error messages were given.

Reordering the walking order is cumbersome. You must point to each field in order, hit Enter, and go on to the next field. If you add one field to the middle of a screen with 50 fields, you'll need to visit 51 fields to include the new field.

Saywhat?! generates only skeletal code. The call to the screen procedure is created as a series of *GoToXY*s. You must add *begin, end,* and a call to the procedure. You don't get compilable source code; the source file must be adjusted any time you change the screen.

No code is generated for editing user input, ranges, or field types, and error messages are not displayed automatically. The Turbo Pascal compiler rejects Saywhat?!'s code because of the dBASE edit masks. In fact, generated code does not even match the examples in the documentation. For my test program, I had to modify my code to produce proper field types to get accurate program and file sizes.

While Saywhat?! supports Turbo Pascal, it has an odd quirk. After you save a screen and leave Saywhat?!, reentry into Saywhat?! returns you to the dBASE default mode. Each time you want to use your screen, you must reselect Turbo. This is a minor, but annoying, flaw.

Saywhat?! could be used as a screen-drawing program, but it lacks too many critical features to be used for prototyping. With no clear way to see field lengths, no edit checks, and no ready-to-compile code, Saywhat?! offers far less than the other products.

The Software Bottling Co. of New York Screen Sculptor

Screen Sculptor from The Software Bottling Co. of New York is a sophisticated prototyping tool that produces compact code with quick turnaround.

After starting the screen editor, you begin a new screen by directly entering text. Fields are defined and background and foreground colors selected with function keys. An arrow key identifies the desired color combinations, as well as their intensity and blinking. This method is also used for the paint mode that allows changing the background color of an area.

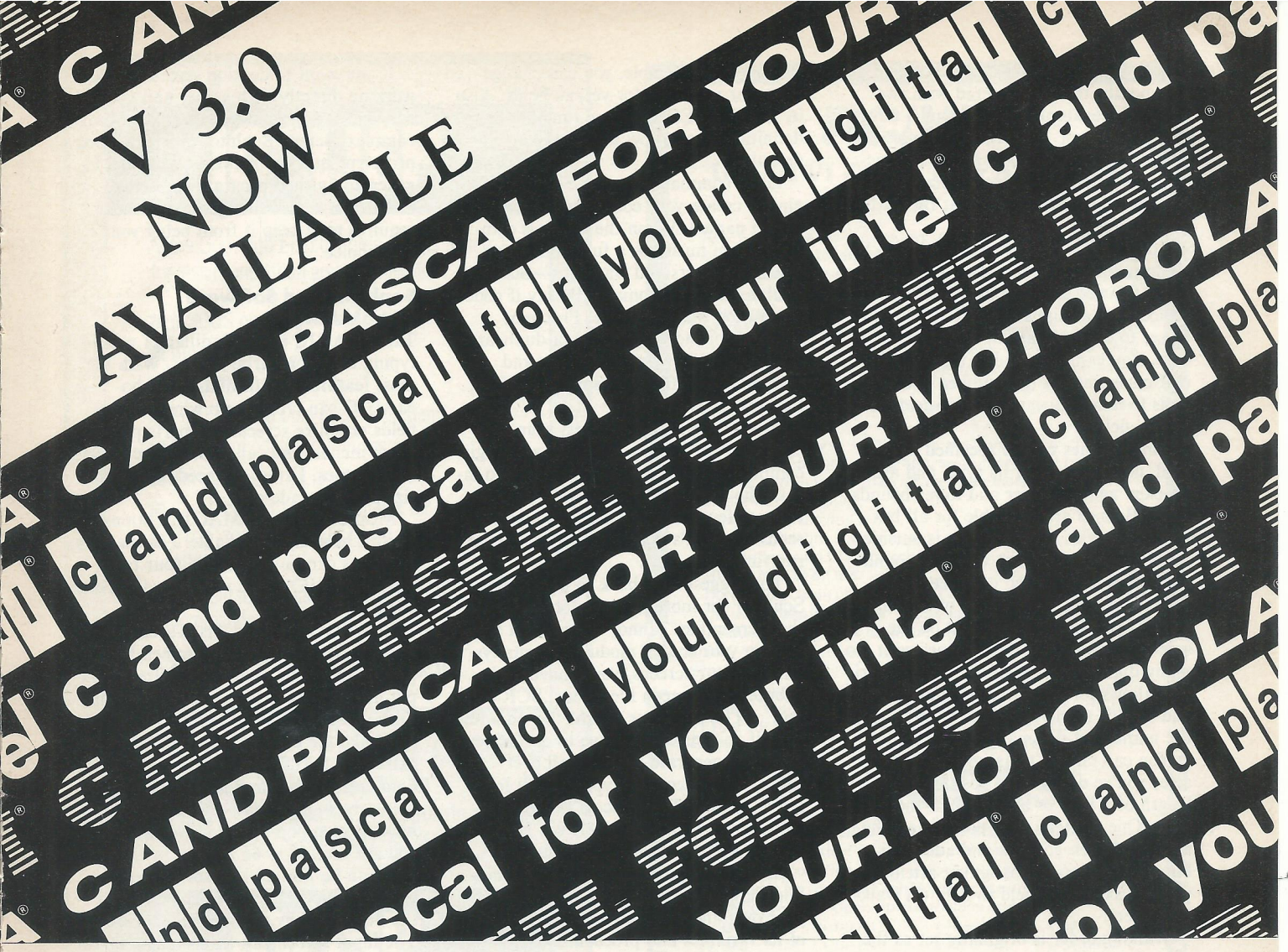
Screen Sculptor supports copying lines or parts of lines. You can't move text or fields, only copy them. As for fields, only complete fields can be copied.

Screen Sculptor can anchor a column on a line. After anchoring at column 40, for example, you can move to column 1 and press the delete key. Only text from columns 1 through 40 is affected—text beyond column 40 remains. Anchoring eliminates the need to pad with blanks after deletions are made.

Graphics characters are selected one at a time from a menu; arrow keys are used to select the desired character. A Control key-direction key combination (left or right arrow, page up or down) draws lines for boxes, but box corners must be selected individually.

Fields are defined with a function key—one strike for each position. A small white box is placed on the screen for each strike; a space must separate fields. Field lengths can be expanded by moving to the end of the field and pressing the function key for each extra position desired.

Attributes for fields include character (strings), numeric (with a selectable number of decimal places), date (MM/DD/YY format only), yes/no,



Whitesmiths, Ltd. Has The Compiler You Want On The Machine You Use.

For over eight years Whitesmiths has focused its efforts solely on developing and supporting a family of quality systems software. Today, Whitesmiths is the only company offering compatible C and Pascal native and cross compilers for the full spectrum of computers on the market—from the IBM PC to the IBM 370, from the DEC Micro-11 to the VAX 8600, and all of the most popular processors in between.

Version 3.0 of Whitesmiths compilers is now available, with new features and functionality designed to meet the needs of today's professional software developers. Features include a C source level interactive debugger with breakpointing and variable display, a powerful multi-segment linker for generating ROM-based code, and the ability to display high-level source code, assembly language, and machine object code on one listing.

With Version 3.0, you get features supporting the emerging ANSI C standard, plus a uniform run-time environment and identical source code across all machine architectures.

The Version 3.0 Pascal compiler provides you with a Pascal to C Translator, a full ISO Level 1 implementation that includes conformant array parameters and numerous extensions.

If you're in the market for C or Pascal, call Whitesmiths at **1-800-225-1030.** We have the compiler you want for your machine.

Whitesmiths, Ltd.

59 Power Road, Westford, MA 01886 • (617) 692-7800 / Telex 750246

INTERNATIONAL DISTRIBUTORS: **AUSTRALIA**, Whitesmiths Australia, P.O. Box 21, 51 Grantham Street, Carlton, NSW 2218, (612) 588-7652 • **FRANCE**, COSMIC S.A.R.L., 52 Quai des Carrieres, 94220 Charenton Le Pont, Paris, (14) 378-8357 • **GERMANY**, GEI, Gesellschaft fuer Elektronische, Informationsverarbeitung MBH, Pascalstrasse 14, D-5100 Aachen, 02408/13-0 • **JAPAN**, Advanced Data Controls Corp., Nihon Seimei Otsuka Bldg., #13-4, Kita Otsuka 1-Chome, Toshima-ku, Tokyo 170, (03) 576-5351 • **SWEDEN**, Unisoft AB, Fiskhamnsgatan 10, S-41455 Goteborg, (31) 125810 • **SWITZERLAND**, RETIS, Realtime Software AG, CH-5001 Aarau, Bahnhofstrasse 96, (64) 247777 • **UNITED KINGDOM**, Real Time Systems Ltd., P.O. Box 70, Douglas, Isle of Man, (624) 26021.

CIRCLE 70 ON READER SERVICE CARD

and male/female. Dates are validated, but months must be entered with two digits (for example, January must be entered as *01*, not *1*). Screen Sculptor also validates sex and yes/no fields. If unacceptable values are entered, Screen Sculptor supplies a prompt at the bottom of the screen.

Edit masks for character fields can be set to permit any character; digits 0-9; digits 0-9 and a blank; or 0-9, a blank, a period, and a minus sign. They can also be set to convert data to upper- or lowercase. You can use a dollar sign in the edit mask and set the number of decimals to two to turn a numeric field into a money field.

Initial values can be specified for any field, no matter what its type, and are checked against upper and lower bounds if set. Bounds are available for all but character fields. If user input is outside these bounds, a prompt with the boundary values is displayed on screen.

Since setting the number of decimal places fixes the decimal point on the screen, users cannot enter values with too many digits. TurboMAGIC, by comparison, allows for a floating decimal point and rounds numeric entries according to the number of decimal places specified.

You can choose to assign attributes after you've defined the position and length of a field or at any point thereafter. One strategy is to position all the fields and then go back and add attributes. Once you are assigning attributes, you can skip from field to field (in walking order).

Walking order is originally set as the order in which fields are created. However, the order can be quickly and simply changed by entering a new field number, and all higher walking numbers will be incremented by one. Screen Sculptor beeps on each field-to-field move, but this can be turned off.

When you save a screen, a .SCR file is created; if fields are present, a .ATR file is also built. After a screen layout is saved, code is generated from Screen Sculptor's main menu for Turbo Pascal, IBM Pascal, BASIC, or Quick BASIC.

You can link up to 12 screens in one test program, a unique feature. Multiple screens are displayed by pressing the page-up or page-down key at execution time, so the order in which the screens are linked in is important.

Screen Sculptor offers you the choice of generating a full program (ready to compile and run) or just skeletal procedure code. You can modify the include file TURPROCS.TUR, which controls many of the functions Screen Sculptor will use. For example, you can change the error messages for yes/no fields. Also in TURPROCS.TUR are function key definitions that allow changing key defaults.

Upon command, Screen Sculptor begins to create your code, checking each screen named. When the check is completed, you must use a function key to check for errors and warnings—Screen Sculptor won't automatically say if code generation was error free. Problems checked for include the use of duplicate variable names between screens and missing screen files (missing .ATR files, for example).

If you create complete test source code, just exit Screen Sculptor and begin compilation. (There is no DOS exit to invoke the Turbo compiler.) Screen Sculptor uses few include source modules and reads a screen's .SCR and .ATR files at run time.

The biggest drawback to using Screen Sculptor for more than prototyping is that screen files cannot be included as code in your source module. If your system has four screens, you'll have to distribute eight screen files (four .SCR and four .ATR) and your .COM file. In addition, if you make a change to one screen, you must generate the code all over again, which requires typing in all four screen names. This process becomes tedious if you want to quickly see how many small changes might look.

Screen Sculptor's documentation outlines how to merge generated code with your own and explains briefly what some of its variables and procedures do. There's no index.

Screen Sculptor's demo disk can give you a feel for how the program works but not for the quality of the generated demo code. The code uses lots of flags and switches, making it difficult to alter without careful study. However, all Screen Sculptor variables end with *SS*, so they're easy to locate.

The help function is easy to use. While the manual doesn't give many examples, there isn't any need to—Screen Sculptor's design is intuitive. Even inexperienced PC users should be able to use the program to layout the text and fields, leaving edit masks and test code generation to a more experienced programmer.

I did discover one bug. In designing a screen for my test program, I was able to create a one-column integer but tell Screen Sculptor the field had three decimal positions. This caused all sorts of error messages, and I couldn't exit the program without rebooting.

Screen Sculptor's biggest advantage is the compactness of the code it produces. Its edit masks are not the most sophisticated or complete, but the turnaround time for viewing contemplated changes is short.

If you want to develop screens while working directly with a user and your edit and logic requirements are not too complex, Screen Sculptor creates a good prototype quickly. However, as a tool to generate finished program code, the number of screen files your application requires may keep it from being your best choice.

Sophisticated Software TurboMAGIC

TurboMAGIC is the Cadillac of prototyping tools for Turbo Pascal. With the most features and flexibility, TurboMAGIC can create help windows for fields or entire screens, supports pull-down menus, and can handle special validation rules. However, speed and code size are the trade-off.

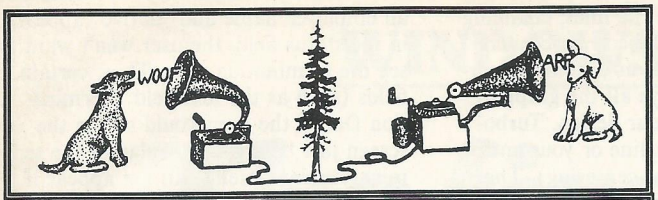
Most of the TurboMAGIC options described here are modifiable. For example, edit keys can be tailored to your taste; TurboMAGIC comes with WordStar-like commands.

Like the other products, TurboMAGIC provides a blank screen to paint. Text fields are entered directly; input and display fields are created with TurboMAGIC's menu system.

Unlike the others, TurboMAGIC is extremely flexible. You can create a data-entry screen and then add a pop-up help screen. Each data-entry field can have its own pop-up help screen, though you are limited to 66 screens in all. Each field can be limited to a set of possible entries, which can be displayed within the field itself by using a selection key. The input screen can be used as a report layout, sending information to the printer, screen, or disk file. TurboMAGIC allows pull-down menus that are similar to those on the Apple Macintosh.

Before discussing some of the more powerful features, a description of basic field attributes is in order. All system control is handled through a pull-down menu system, although some Control-letter combination commands are available.

The basics of a field definition are the field's Turbo Pascal field name, its use (such as input or display), justification (left, right, centered, or none), the fill character, whether to make the input uppercase, whether to allow blank input (only available for certain field types), video settings (if different from a form's default values), upper and lower bounds (if any and for numeric fields only), and the picture mask. Colors can be selected for the entire screen (called a form) or for individual files. TurboMAGIC allows for characters such as parentheses to be entered in the edit mask, and generated code will skip these positions during data entry.



WOOF to ARF ???
Another translation by QPARSER+

Dogs don't use trees for translation, but with QPARSER+, you can use trees to develop compilers, interpreters, complex user-interfaces, language and file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc..

QPARSER+ -- don't develop a language-intensive program without it !!

- Companion college text book on compiler design available.
- A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over our first Translator Writer called QPARSER.
- Widely used in both industrial and educational settings.
- Sample translators included: Pascal, Assembler, etc.
- Variable level debugging and extensive error recovery.
- Full LALR(1) parsing strategy used.
- Runs on 512K PC/XT/AT or VAX/VMS
- Generate full translators in C, Pascal, or any other language.

QPARSER+ -- just $475 (PC/XT/AT)
Call or write for details and a FREE demo disk

QCAD Systems
3333A Octavius Dr, Santa Clara, CA 95054
(408) 727-6671-- outside of California call (800) 538-9787
dBase III © Ahston-Tate; Word © Microsoft; Wordstar © Micropro

CIRCLE 73 ON READER SERVICE CARD

A PROGRAMMER'S TOOLBOX

screen master®
Bridging management's dreams and programming's genius

ONLY $99.95

A DP MANAGER'S BEST FRIEND

PURE GENIUS IS NOT ENOUGH
(YOU STILL NEED THE RIGHT TOOLS)

- DESIGN MENUS
- CREATE PROTOTYPES
- CREATE TUTORIALS
- CAPTURE SCREENS
- CREATE DEMOS
- RUN TIME MODULE

23 EXCITING NEW UTILITIES FOR PROGRAMMERS!

ENHANCE HIGH LEVEL LANGUAGES WITH FULL ACCESS TO ALL CAPABILITIES IN YOUR OWN CODE

"source code was reduced by one third..."
"15 minutes to design a sophisticated screen..."
Scott McCaffrey, Musco of PA

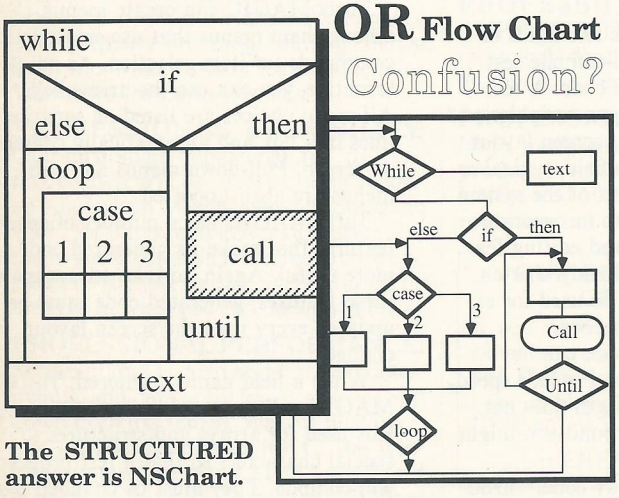
"In a word, fantastic..."
"...I just returned my copy of Dan Bricklin's Demo Program." Thomas Emr, Dir. of Marketing, ADP Inc.

(717)652-1200
GENESIS DATA SYSTEMS

TO ORDER SEND $99.95 plus $3.00 S&H TO: GENESIS DATA SYSTEMS - 5403 Jonestown Rd. Harrisburg, PA 17112

CIRCLE 71 ON READER SERVICE CARD

NSChart Structure OR Flow Chart Confusion?



The STRUCTURED answer is NSChart.

NSCharts are directly generated from text files. No symbol manipulations are required. Text files consist of NSPDL, a Siltronix structured design language.

Translators to NSPDL from FORTRAN, Pascal, C, SDDL, and Caine-Farber-Gordon PDL are also available from Siltronix.

NSChart runs on Macintosh and IBM PCs. Both feature "What You See Is What You Get".

from SILTRONIX
P.O. Box 82544 San Diego, CA. 92138
Sales Info 1-800-637-4888 Technical Support
Macintosh is a trademark of Apple Computer Inc.
IBM is a trademark of International Business Machines Corporation

CIRCLE 72 ON READER SERVICE CARD

LOTUS TO C
The Power of LOTUS with the Flexibility of C

The LTS__2__C™ translates the entire LOTUS worksheet, Data and Formulas to C Language source file ready to compile on PC Mini- or Mainframe.

- Works on LOTUS releases 1 and 2
- Library of all spreadsheet functions
- Compatibility with most C Compilers
- Fast, portable, executed in one command

RESOURCE ANALYSIS INTERNATIONAL (RAI)
12581 Venice Blvd.
Los Angeles, CA 90066
(213) 390-7661

*LOTUS is a registered trademark of LOTUS DEVELOPMENT CORPORATION.

CIRCLE 74 ON READER SERVICE CARD

You can restrict entered data by using picture elements (numeric only, numeric and spaces, and so on). The picture element choices are adequate, and new ones can be added. These new picture elements are then available to all your source code. Unfortunately, a field's mask specification is not correctly explained in the documentation, and the screen notation is nonstandard, adding to the confusion.

An individual help message can be entered when defining a field's attributes. The message will automatically appear on the bottom line when the cursor moves to that field. If a pop-up help window is to be used for the field, it must be created as a different file and then incorporated into your main source code; it can be any size up to a full screen.

TurboMAGIC offers the most field edit masks: single character or string, date, numeric, social security, telephone, zip code (both five and nine digit), true/false, male/female, yes/no, integer, real, and dollar amount (with or without asterisk fill). TurboMAGIC supports state (all 50 and the District of Columbia), hour:minute (HH:MM format), and month. TurboMAGIC also supports real numbers with incrementation. This type allows you to set a initial value; the user can increment that value up or down by the amount set in the increment field.

When entering data in an integer field, a user will get an error message if the value entered is out of bounds (-32,768 to 32,767). If decimals are used, TurboMAGIC will round a number that has too many decimal places. For example, if 1.259 is entered, 1.26 is stored.

The menu field type can be used to limit entered data to specific values. While this is a clever feature, it's awkward to use to select a value. When you reach a field of this sort, the default value is displayed. If you want to change the value, you have to go through the alternative values until you reach the desired one, rather than typing in the desired value. The alternative is to create your own logic, which means changing generated code each time you change a screen.

When the field has been entered, TurboMAGIC asks you if you want to exit the form. If you don't, you'll loop through all the screen's fields until Escape is pressed. TurboMAGIC allows for 66 lines per screen; you hit the Page Up or Page Down keys to jump between pages of the screen.

The TurboMAGIC editor can draw single- and double-line boxes by changing the arrow keys to box mode. Moving the arrow keys draws the lines; changing directions results in the correct corners (sometimes referred to as automatic corners). A menu displays all the graphics that can be used in your design. TurboMAGIC can center a line or your entire layout on screen (page centering). The program allows for copying, moving, and writing blocks to a text file.

Help screens and screen layouts can be incorporated as procedures into the main source file or read as picture files at run time. If they are run as procedures, the main source file must be changed; the generated source shows you the spaces left for the procedure. After you generate the help screen procedure code as an include file, you must remove the stub TurboMAGIC has created in the main source program and substitute that file for the help procedure. TurboMAGIC points the way for this change with instructions in both the main module generated source and the documentation.

Positioning for help windows is automatic but can be specified if desired. If the help text is too long for a help screen, text can be scrolled up and down within the frame.

When using picture files, if there is more than one screen, you must link them together (for example, a screen plus a help screen). TurboMAGIC calls this collating. Happily, TurboMAGIC will search your main program and determine what files should be incorporated. Further, this list of files can be saved, so you need not repeat the procedure for minor changes.

TurboMAGIC creates two kinds of code: complete, ready-to-compile test programs and procedure code. A test program defines all its own variables and can give users an idea of screen layout, cursor flow from field to field, typical error messages, and the feel of the system. Procedure code is used to incorporate windows, help screens, and editing features into code you've already written. The procedure code is also used for external picture files for screens.

However, generated code can be lengthy. Even with Turbo Pascal's speed, TurboMAGIC's code length does not provide the rapid turnaround you might want for prototyping.

In exchange for lengthy code, TurboMAGIC does offer powerful options. For example, fields can be conditional, meaning that they're displayed or available for input only under stipulated circumstances that you code separately. Such logic must be added to the generated code, but TurboMAGIC does make note of where such code belongs.

Another useful option is the prompt field, a text field that is shown on the basis of the conditional field that precedes it. For example, if you enter a request for an employee name and "active" appears in the status field, the user won't want to see the termination date. Thus, certain fields (such as the text field "Termination Date:" the user would see on the screen just before the display of the termination date itself) will not appear if the status field (conditional field) contains "active." This design makes for a cleaner screen, but again such fields call for code modification every time screens are changed.

Two types of dependent fields are allowed: value dependent and use dependent. Value dependent means that dependent fields change value when a control field changes. For example, if an order subtotal changes, then the sales tax amount must be changed. Sales tax is the value-dependent field since it is recalculated when the order subtotal changes.

Use-dependent fields control the appearance of other fields: If *Sales Tax?* is *N*, the *Sales Tax Amount* field does not appear for display or entry. Dependent fields can be dynamically updated, with updating occurring for all dependent fields as soon as the cursor leaves the control field.

Dependent fields mean extra coding outside of TurboMAGIC, and such code must be reincorporated any time you make changes to screen layouts. TurboMAGIC will point to the areas of code that need to be changed, but updating is left to you. This lengthens the time between changing the screen and viewing results.

TurboMAGIC can create menus, among them menus that use single-letter commands for item selection. As an alternative, you can use the arrow keys. All menu choices are listed on separate lines in a box and automatically centered on screen. Pull-down menus and submenus are also supported.

TurboMAGIC has a number of special features that make its generated code more useful. Again, to take advantage of these features, generated code must be updated every time the screen layout changes.

When a field name is entered, TurboMAGIC will convert brackets and periods used for arrays and structures to special characters so that a test program will compile. They must be changed back later, if desired.

Just about everything in TurboMAGIC can be customized. An installation option lets you check for snow and permits customization of code to fit a monitor's characteristics.

An installation program allows users to choose printer and screen options. This program modifies the TurboMAGIC

TP2C™ C CODE FROM *TURBO PASCAL

YOU NEED *TP2C*™ IF:

- You Like the ease of Turbo Pascal but wish to create linkable libraries.
- Your Turbo Pascal programs are too large for COM files.
- You want Pascal strings and sets in C.

TP2C™

* Produces standard C code DIRECT from Turbo Pascal programs.
* Create function libraries from Pascal routines.
* Supports strings, sets, and Pascal I/O.
* Pascal programmers may use as an aid in learning C.
* Move Turbo Pascal programs to Unix or other machines.
* Translate whole programs or program segments
* For PC-DOS and MS-DOS machines.

Order Today only $249.00

BISS of Louisiana
5432 Superior Drive,
Suite A-1,
Baton Rouge, LA 70816
(504) 295-0260

*Turbo Pascal is a registered trademark at Borland International

CIRCLE 75 ON READER SERVICE CARD

ACCELERATE YOUR C DEVELOPMENT

RTC PLUS

FORTRAN/RATFOR TO C TRANSLATOR*

- **Maximize the vast resources of FORTRAN** while moving up to C. Speed up new C development and avoid reinventing the wheel.
- Use ***RTC Plus*** to translate FORTRAN code and libraries — and **maintain code with greater ease and flexibility** in C.
- **Source code to C libraries is included.**
- ***RTC Plus* supports standard FORTRAN-77** as well as some DEC VAX extensions (excluding FORTRAN I/O, character and complex statements/expressions). Over 95% of STUG's RATFOR is supported. The Translator generates K&R C.
- **Finally a cost-effective method of conversion into C.**

SPECIAL OFFER!

DEMO $10
MS-DOS $325
Includes FREE source code control system disk!
(Offer good until July 31)

***Translate: "To convey to heaven without natural death."**

COBALT BLUE
1683 MILROY, SUITE 101, SAN JOSE, CA 95124
408-723-0474

CIRCLE 76 ON READER SERVICE CARD

TAKE COMMAND.

INTRODUCING THE SURE-FIRE WAY TO TAKE COMMAND OF **MS-DOS**
INTRODUCING COMMAND PLUS ONLY $79.95

Now you can take command of MS-DOS with Command Plus, the programming shell and command processor that replaces COMMAND. COM in Versions 2.0 and above.

For example, Command Plus' aliasing function lets you create fast, memory resident macros, while the history command lets you use the cursor keys to recall, edit and execute commands that you ran up to 48 command lines ago.

What's more, Command Plus also equips you with Script, a batch processor that's easy to learn and unbelievably powerful.

In short, Command Plus means that you can now program easily and efficiently in the familiar DOS without having to learn UNIX.

At only $79.95, plus shipping and handling, the price of Command Plus is pretty neat, too.

So, if you want a sure-fire way to take command of DOS, or if you want more information about Command Plus, call us at (800) 992-4ESP. In California, call (213) 390-7408. VISA and MasterCard accepted.

(800)992-4ESP (213)390-7408

COMMAND PLUS.

ESP SOFTWARE SYSTEMS INC.
11965 Venice Boulevard
Suite 309
Los Angeles, CA 90066

ESP

MS-DOS is a registered trademark of Microsoft Corporation

CIRCLE 77 ON READER SERVICE CARD

Bpcs

Breakpoint Computer Systems, Inc.

Introducing: QMan, a true relational data base management system which supports both interactive and imbedded queries. Based on the IDL query language from Britton Lee Inc. A great foundation for applications, and an incredible package in its own right. Features include: multi user access, B+ Tree Indices, imbedded queries, sorting, and sort merge. Portable to most SYS V machines. Include machine name when ordering. **Just $895.**

QMan now available for DOS: $219

Also available through Bpcs: from Microport Systems System V/AT: UNIX™ for your AT!

	List	Bpcs
RunTime System	$199	**$179**
Development System	$249	**$225**
Text Processing System	$199	**$179**
Complete System V.2	$549	**$485**
Unlimited Upgrade	$249	**$225**

Breakpoint Computer Systems, Inc.
6701 Seybold Road, Suite 204
Madison, WI 53719

VISA and Mastercard accepted. For Microport products add $6.00 per single package or $14.00 complete system.
UNIX is a registered TM of AT&T

Call collect
(608) 271-2171

CIRCLE 78 ON READER SERVICE CARD

New Release!

SEIDL MAKE UTILITY

Version 2.0

The BEST just got BETTER!

If you're serious about software development, consider the **SEIDL MAKE UTILITY** (*SMK*). *SMK* is not just another copy of the Unix Make. It was specifically designed to deliver features and performance not found in other makes.

✓ ***Structured Language*** to describe dependencies in a clear, concise and portable manner.

✓ ***Rich Command Set*** includes parameterized macros, variables, if-then-else, iteration, wild cards, exception cases, macro libraries, interactive statements, environment access, pattern matching and much more!

✓ ***Intelligent Analysis*** algorithm handles nested include files, library dependencies, and performs consistency tests to detect errors that other makes would blindly ignore.

✓ ***Seidl Version Manager*** compatibility lets you expand your system into the most comprehensive revision/version control system available.

"SMK is a very good Make indeed. Its major distinction is a truly simple Dependency Definition Language, easy to learn and easy to use... you'll probably bless SMK."
— Sextant, July '86

"SMK offers many unique features. [The error handling facility] is extremely useful if a large number of files must be recompiled."
— Computer Language, June '86

DOS Version Only **$99.95** $3.50 p&h Call for other op systems.

Call Today
1-313-662-8086
Visa/MC/COD Accepted
Dealer Inquiries Invited

SEIDL COMPUTER ENGINEERING
3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE 94 ON READER SERVICE CARD

.COM file by looking for constants within the file. The 8,300-line, 59K program requires two additional files be distributed with your source code (one for printers, one for on-line help).

TurboMAGIC includes three programs in .COM form only: a program printer with options for capitalization, page numbering, and expansion of include files; a mortgage finance program that demonstrates several help, menu, and field principles; and a multiple-directory file find.

TurboMAGIC clearly offers the greatest variety of prototyping options. Given the number of overlays, it performs quickly and handles some of the data-entry chores in an intelligent fashion.

While TurboMAGIC offers many options, you'll have to decide whether to use them. While they're clever, dependent fields require modification every time you make a screen change. Since the goal of prototyping is to create revised working models quickly, the generation of new source that requires little modification is critical. The key criteria are how fast your changes can be implemented and the size of the resulting program.

TurboMAGIC's strength is program generation. As for shortening turnaround time, updates to source code can be minimized with include files. However, logging modifications and annotating source can often fall through the cracks.

The TurboMAGIC manual offers some suggestions on incorporating changes into your source. The appendixes describe ways to slim down source code, modify TurboMAGIC's include code for function or editing key selection, and the input procedures used in generated code.

The documentation suffers from too few examples in early chapters; a tutorial would be helpful. By the time you reach the first example, there are just too many terms to keep them all straight. This is another price you pay for a complex program.

When using the documentation, I sometimes had the annoying feeling that it wasn't originally written in English. "If help window were not provided in old version..."—Ugh!. Also, the printing makes i's look like l's, which is distracting.

On the whole, TurboMAGIC does what it claims it will do. The generated source code and its documentation are simple enough to be understood. The number of options available and the product's terminology do take some getting used to, but learning them shouldn't unduly delay any project.

Hawaiian Village Software Turbo Master

Turbo Master from Hawaiian Village Software takes Screen Sculptor and goes it one better. Its screen-paint program offers virtually the same features as Screen Sculptor's, with some nice extras: undo of insert and delete lines, generating a screen dump report or a report on screen variables, and limited macro capability.

Turbo Master also supports changing the border color and provides state edits. Date and time field types can be automatically initialized to the system date and time. (However, it does not offer upper and lower bounds for dates.)

Turbo Master's approach is to use a data base of fields. You paint the screen with field placeholders, but field type, upper and lower bounds (for numerical variables), and masks are kept in a data base. Elements in the field data base are then available to the ISAM-generator, which I'll describe later. This design has one limitation: if two screens contain the same field for entry, their upper and lower bounds must be the same.

New data types, which are called character sets, can be created. You can specify up to four ranges of single characters (A through L and Q through S, for example) or use their decimal equivalent. You can also use a set of characters (A,E,I,O,U, for example) containing up to 18 elements. These character sets are turned into Turbo Pascal sets (['A','E','I','O','U']).

Turbo Master is so similar to Screen Sculptor that it provides for format conversion for those screens. (It converts text and graphics but not field definitions.) Several screens can be tied to a group, so, unlike Screen Sculptor, Turbo Master will regenerate all screens with only the group name.

The program lets you specify before and after procedures for each field. When you name a procedure you want to use to pre- or postedit an input field, Turbo Master will place procedure stubs with comments in your code. You can use .INC files to replace these stubs.

Help screens for fields are supported, but there's a limit of nine help screens per screen. The help screens are designed with Turbo Master and must be full-screen size. Each field can have a single-line prompt, but they're displayed only after the user has pressed the first key. (This makes no sense—a prompt should appear on entry to a field.)

Turbo Master can also generate the ISAM file layouts using the file data base and field descriptions from your screens. After defining up to four keys and the screen file name or group name, you're asked for key length, page size, and information for other parameters of Borland's Turbo Database Toolbox. If you want to use Turbo Master's ISAM procedures instead, this screen should be skipped.

To use Turbo Master's ISAM procedures, a memory-resident program must

SAPIENS V8

A VIRTUAL MEMORY MANAGER FOR THE PC

Sapiens V8 is a virtual memory manager for C programmers on the IBM PC. The unique design of the virtual memory paging storage scheme makes it an invaluable tool for C programmers. With an 8 mg. workspace and stack and heap libraries Sapiens V8 solves the problems and hazards of stack and heap overflow.

Sapiens V8 brings the power, speed and complexity of a mainframe virtual memory manager to the PC. Sapiens V8 virtual memory manager emulates a 80386 memory management system and is fully implemented in software. It is not dependent on add-on boards or special chips.

- **Provides 8 mgs. virtual memory workspace**
- **Fast: less than 10% speed overhead**
- **Advanced software emulation of 64-bit architecture**
- **Virtual stack and heap: multiple return values, stack frame management. (vmalloc( )).**

System requirements: Huge model C compiler (Microsoft, Lattice, Aztec). V8 uses 22-128 Kb core. Hard disk recommended.

LIST: $300.00 (408) 458-1990

Sapiens Software Corporation, P.O. Box 7720, Santa Cruz, CA 95061

CIRCLE 79 ON READER SERVICE CARD

CANADA'S SOURCE FOR C

– Canadian Sales
– Canadian Service
– Canadian Technical Support
– Canadian Product Knowledge

We specialize in programming & development software

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX
SOFTCRAFT • MICROSOFT • BLAISE • ESSENTIAL
AGE OF REASON • DESMET • AZTEC
MARK WILLIAMS • GIMPEL • ROUNDHILL • GSS
HALO • FAIRCOM • RAIMA • INTEL • etc. • etc. •

VISA Call for full price list—Dealer enquiries welcome MasterCard

We know our products — we use them!

SCANTEL SYSTEMS LTD.

801 York Mills Rd., Don Mills, Ont., M3B 1X7
(416) 449-9252

CIRCLE 80 ON READER SERVICE CARD

Modula-2

IBM PC/DOS
Native Code Compiler

This is a full implementation of Niklaus Wirth's Modula-2 language. Our product is not an interpreter, but a true 8086 compiler, using state-of-the art techniques. A Unix-like "make" utility is included which provides automatic recompilation of modified source programs.

The code generator produces object module input for the DOS link utility. You may combine your Modula-2 programs with code from other languages such as Assembler. The software also operates on PC compatibles using MS/DOS. All the run time source code is included. None of the software is copy protected, and is fully supported and maintained by farbware. No royalties are charged for the use of the run time object. A complete and comprehensive reference manual is included in the purchase price. The manual is available separately for $25.00.

Site licenses and quantity discounts are available.

$89.95 Complete

farbware
1329 Gregory
Wilmette, IL 60091
(312) 251-5310

Master Card and Visa Accepted

CIRCLE 81 ON READER SERVICE CARD

Parallel Programming for "C"

INTERWORK

A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX*-style signals. Also has building blocks for constructing your own ITC facilities.
- Handles interrupts (DOS version) and integrates them into task scheduling. Supply your own interrupt handlers or block tasks on interrupts.
- Lets you trace task switches and inter-task communication.
- Comes with complete documentation including a user's manual and reference manual of commands.

Interwork is available for the following systems:

Hardware	*Operating System*	*Price*
IBM PC, XT, AT	PC-DOS 2.0 or later	$129
IBM PC AT	XENIX*	$159
DEC VAX*, SUN	UNIX 4.2BSD	$249

PC-DOS version is compatible with DeSmet, Lattice, and Microsoft C compilers.

Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add $2.50. Send check or money order to:

Block Island Technologies
Innovative Computer Software
13563 NW Cornell Road, Suite 230, Portland, Oregon 97229-5892
(503) 241-8971

*Trademarks: UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

CIRCLE 82 ON READER SERVICE CARD

NEW FROM
C WARE

TA-DAH!

DESMET C
V 3.0

THE FASTEST
C COMPILER JUST GOT
FASTER!

FASTER DEVELOPMENT

Invoke DeSmet C from our SEE[tm] editor. The first error returns you immediately to SEE at the error line with the error message displayed.

FASTER COMPILATION

When you don't use inline assembly code or don't want to see the ASM88 output, the V3.0 compiler produces object code directly – making DeSmet C up to ***twice*** as fast as before.

PLUS

EXPANDED STANDARD LIBRARY

New environment, networking, path, file, time and string functions now included.

DESMET GAZETTE

While our supply lasts, we are shipping all new customers an introductory copy of this excellent code intensive quarterly published by Pacific Data Works.

C88 C Compiler Still only	$109
D88 Debugger	50
Large Case Option	50
MAKE utility (an excellent value)	50

Product literature sent on request.

Call us now and we'll ship today.

C WARE
CORPORATION
(408) 720-9696 TELEX: 358185
P.O. Box C
SUNNYVALE, CA 94087 USA
VISA, MC & AMEX
Street Address: 505 W. Olive Suite 767
Tech BBS (408) 720-8197
300-1200-2400 BAUD

be loaded that takes up 77K. The documentation claims the program eliminates 8K of code space required for Borland's Turbo Database Toolbox. While I didn't check this, I can say that file creation and look up is speeded up. Turbo Master generates Btrieve code flawlessly.

The generated program uses the screens to access the data base. If the search is not successful, the program allows you to add the record. The integration of screens and the data base is good, though the main menu it generates suffers on monochrome monitors. It has too many background colors, which are distracting and cause eye strain.

A skeleton report is generated with a simple header and footer line and automatic page numbering. It prompts for whether it should write to the screen or the printer. The generated report is well documented, easily modified, and lists the data base's key field.

Turbo Master can produce a menu system that uses highlighted capital letters for item selection. You list the procedure name and the .CHN or .COM file to execute. The generated code requires that you add the procedures or calls to the .CHN or .COM files.

The package does not automatically adjust for monitor type. Its shading choices for monochrome are difficult to read when using a color card with a monochrome monitor. The menu is especially difficult to read; you won't want to work with the program for long periods of time. With a color monitor, the screens are too busy with colors.

The documentation lacks an index; the small, dark type makes reading some of the text difficult. The documentation describes Turbo Master's procedures, including screen reads and writes, string manipulation (uppercase, justification, etc.), and date and time validation. The appendixes contain dozens of pages about how to modify defaults, incorporate your own code, use windows, create relational data bases, and change the walking order.

Turbo Master does not buffer its keyboard, so there's a lot of waiting for programs to finish. The Escape key is used to continue, not the return key, a somewhat unnatural choice. Deleting characters is slow.

Turbo Master has other deficiencies. The help key didn't work on some screens. The naming conventions are restrictive. The program uses lots of overlays, and the generated program code also contains many overlays.

Error messages don't make clear which file extension the program is having trouble with. Messages appear with no hint as to whether they are warnings or critical errors. When I created ISAM code before the ISAM data file had been created, a message said I was missing the ISAM file. This apparently meant there was no ISAM data file rather than no ISAM program source file. The documentation doesn't explain the program's messages.

While the documentation describes most of the file-naming conventions, it fails to mention that you'll need extra files on any distribution disk you make—.COM and .SCR files are not sufficient. The number of files you'll need is worse than the number required by Screen Sculptor. I needed my .COM file plus three overlay files plus the .SCR file. The .COM size in Table 1 on p. 139 reflects the total number of bytes for all but the .SCR file.

The manual instructs you to change your CONFIG.SYS file to BUFFERS=50. Unfortunately, this wreaked havoc on my Zenith, causing the program to display the wrong time, fail to properly edit male/female fields, and display "Cannot read Partition Table" on boot up. (For a moment I thought the program had somehow crashed my hard disk.) Setting BUFFERS=20 solved the problem. The program also has the annoying habit of changing the DOS system time.

Jim Powell is a microcomputer support analyst with Holland America-Westours in Seattle, Wash. His interests include Turbo Pascal, C, and microcomputer file systems.

Pascal products and manufacturers

Saywhat?!—$49.95
The Research Group
88 S. Linden Ave.
S. San Francisco, Calif. 94080
(800) 468-9273
(800) 231-7849 in Calif.

Screen Sculptor—$125
The Software Bottling Co. of New York
6600 Long Island Expwy.
Maspeth, N.Y. 11378
(800) 872-8787, operator 10 or 11

TurboMAGIC—$99
Sophisticated Software Inc.
6586 Old Shell Rd.
Mobile, Ala. 36608
(800) 225-3165
(205) 342-7026 in Ala.

Turbo Master—$69.95
Hawaiian Village Computer Software
1109 Pennsylvania Ave.
St. Cloud, Fla.
(800) 821-9503
(800) 342-0137 in Fla.

Editor features

Manufacturer and Product	Insert/delete line	Block move	Move a field	Line move	Undo	Anchor[1]	Field types supported: Numeric	Date	Male/female	Yes/no	Character (string)	Upper/lower bounds checking	Display/input distinction	Automatic exit control	Automatic box corners	Largest repaint area
The Research Group Saywhat?!	no	yes	yes	yes	yes	yes	no	no	no	no	no	no	yes	no	yes	B
The Software Bottling Company of New York Screen Sculptor	yes	no	no	yes	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	S
Sophisticated Software Inc. Turbo MAGIC	yes	yes	yes	yes	no	no	yes	yes	yes	yes	yes	yes	yes	yes	yes	B
Hawaiian Village Software Turbo Master	yes	no	no	yes	L	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	S

B = Block
S = Single square
L = Limited
1. Anchor feature helps with deletions.

Table 1.

General features

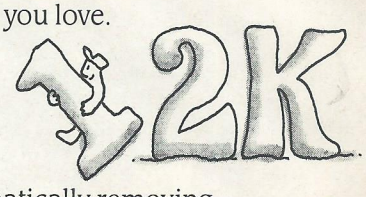
Manufacturer and Product	Screen type: Separate file	Pascal procedure	Creates test system ready to compile	Test program: Total lines compiled	Size of screen files	Change skeletons or INCLUDE files	Adjusts for monitor type	Documentation explains code	Generates help windows	Generates menus	Exit to DOS for compilation	Other languages supported	Display screen image from DOS
The Research Group Saywhat?!	yes	yes	no	42	784	no	yes	yes	yes	no	yes	yes	yes
The Software Bottling Company of New York Screen Sculptor	yes	no	yes	898	4,296	yes	yes	yes	no	no	no	yes	no
Sophisticated Software TurboMAGIC	yes	yes	yes	4,774	3,872	yes	yes	yes	yes	yes	no	no	no
Hawaiian Village Software Turbo Master	yes	no	yes	4,695	3,968	yes	no	yes	yes	yes	no	no	no

Table 2.

THE OPTIMIZER. TAKE TURBO TO THE LIMIT.

Turbo OPTIMIZER™ helps you squeeze down, rev up, and fine tune your compiled Turbo Pascal programs. Now there's no need to write in assembler...you program faster and easier in the language you love.

A LEANER TURBO

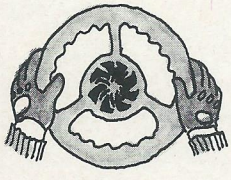


Small programs of 12K bytes are often reduced to only 2K. Medium-sized programs can shrink by 7K bytes.

The Object Code Compactor makes code smaller by automatically removing the unused portions of the run-time library as well as unused procedures and functions in your own code.

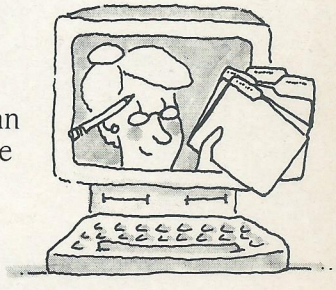
You save valuable disk and RAM space.

A FASTER TURBO



The Object Optimizer scans the object code and removes unnecessary instructions like null jumps and register reloads. Your Turbo Pascal programs run up to 30% faster.

CALL ON THE LIBRARIAN



Store compiled versions of your often-used procedures in relocatable form using The Object Librarian. The Object Librarian then creates External procedures that you include in your source code. You cut compile time.

"A real lifesaver...I saved 10K bytes in the latest version of Note-It®"
Dave Whitney, author, Note-It

Call or send today for Turbo OPTIMIZER. It's only $75. (With source code, $125) Call toll free for credit card orders:
(800) 538-8157 x830
(800) 672-3470 x830 in CA.
Satisfaction Guaranteed or your money back within 30 days.

For other information call (408) 438-8608. Shipping and taxes prepaid for U.S. and Canadian customers. Others please include $6 per item for shipping.

More ways to take Turbo to the limit:
T-DebugPLUS, the symbolic run-time debugger, only $60,
Turbo EXTENDER breaks the 64K barrier, only $85,
TurboPower Utilities, nine powerful programs, only $95 with source.

TURBO Power

TurboPower Software, 3109 Scotts Valley Dr., Suite 122, Scotts Valley, CA 95066
© 1987 TurboPower Software
These products require Turbo Pascal 3.0 (standard, 8087, or BCD), PC-DOS 2.X or 3.X, and IBM PC/XT/AT or compatibles.
Note-It is a registered trademark of Turner Hall Publishing, a division of Symantec.

CIRCLE 87 ON READER SERVICE CARD

SOFTWARE REVIEWS

New tools: COBOL screen management

By Chuck Ballinger

You purchased your COBOL compiler months ago and have been busy coding ever since. One day, it suddenly hits you: you're spending the largest percentage of your time not actually programming but in laying out, designing, and coding your input/output screens.

Take heart! COBOL screen managers are now available to help with this tedious task and greatly increase the time available for design and program coding. You'll also find that these tools can make your COBOL programs appear as if they were written in C or Pascal. With them, you'll have access to features that until now were just not practical for the COBOL programmer. These new features include windowing, context-sensitive help screens, graphics, and color—all with minimal effort on your part.

This review covers three COBOL screen managers: Master Computer Systems' FORMIX, Northern Computing Consultants' SCREENIO, and Flexus International's Screenplay. Table 1 offers a quick comparison of the products and the languages supported. While this review covers the products' screen management abilities with respect to COBOL, note that their support of other languages varies.

Unlike compilers, screen managers cannot be benchmarked in the true sense of the word. Instead, they must be evaluated for their functionality and usefulness. All three were tested on an IBM PC/XT with 640K RAM, two 10MB hard drives, a color graphics adapter, and DOS v.3.1.

Master Computer Systems FORMIX

FORMIX from Master Computer Systems is the most powerful of the packages reviewed. In fact, with over 50 functions it's a challenge just to learn to use it. While not supplied with a demo disk, the product can be picked up in a reasonable amount of time.

The system is written entirely in assembler and is extremely fast during program execution. FORMIX is a run-time system; it doesn't generate source code for screens but places them in a library for access by an application at run time.

FORMIX handles all routine data manipulation (formatting, justification, editing, and validation), a feature the other two do not provide for. FORMIX also handles error messages, help text, and operator messages.

An executive program controls the display screen and keyboard on behalf of the application program. As a program executes, it carries on a conversation with the resident executive through a language interface. The program can then command the executive to perform one of 57 functions.

FORMIX's library files are stored in standard DOS format. A library file can contain these file types: message, keyboard, configuration, and panel (the FORMIX version of a screen). By consolidating these files into a single DOS file, Master Computer Systems has substantially reduced the clutter normally associated with multiple libraries.

The application program loads a panel through the executive using the FORMIX work space. The panel's fields are not displayed until an application activates the panel through another executive function call. More than one panel can be maintained in the work space and used with windowing for quick screen response. As with the other two products, panels can be managed through a panel maintenance program as well as with the full library maintenance routines.

FORMIX's programming support utility takes panel buffer definitions, which are based on symbolic names assigned during panel maintenance, and produces source code for inclusion in a program. Once source code is generated, further changes can lead to compatibility problems down the road.

Since FORMIX does most coding at the transaction level, you'll have to generate more code than with the other screen management programs. This bulk is offset by FORMIX's extensive control over the input and display of panels and program execution.

FORMIX is the only product reviewed that charges royalties for distribution of run-time systems. If you purchase multiple language support modules, each additional module is $295 if purchased at the same time. FORMIX v.2.0 requires an IBM PC or compatible with 256K RAM, DOS 2.0 or later, and a floppy drive; a hard disk is recommended. The memory requirements of the run-time system are dictated by the size of application programs.

Northern Computing Consultants SCREENIO

SCREENIO from Northern Computing Consultants is unique in that it's written entirely in Realia COBOL. Panel images can be stored in a program's *DATA DIVISION* or a separate run-time library.

The package has two principal parts: the panel editor and the SCREENIO subroutine. The panel editor designs display screens in an interactive, WYSIWYG fashion—it will show you the colors, text, and fields exactly as they will look under program control.

SCREENIO generates COBOL source code that's stored in a source library file. You can have separate libraries for each application you develop. The panel editor manages the libraries for you, which allows copying panels between libraries, deleting panels, merging libraries, and the like.

Once screens (panels) are designed, they're stored in a library file or integrated into your code. Happily, source code does not have to be changed when moving panels from a library to a program—just regenerate the panels and then recompile your program with a COBOL *COPY* statement. The only programming required is calls to the SCREENIO subroutine.

With SCREENIO, you have access to over 80 PC function keys and key combinations. Your program not only can tell what keys were pressed but also which field it was in and the relative position of the cursor within the field.

SCREENIO windowing is unique in its supports of multiple open windows

QC

A 'C' Interpreter for the PC

int, char, long, float, struct, union, pointers,
function pointers, pointers to pointers, arrays
Standard C function library - Help windows
Command processor shell - I/O redirection
Full screen editor - Full screen debugger

QC is for the C Beginner

QC is an ideal learning tool. Enter and run the exercise programs in the QC reference manual or those from many of the popular C books. Your learning process is enhanced by QC's fast execution and interactive debugger. Watch your program step through the source code. Set breakpoints. Examine and alter variables. Execute C expressions from the keyboard.

Learn at your own pace - don't be bound by a slow compiler and linker.

QC is for the C Veteran

QC is an excellent tool for the professional programmer. Use the QC full screen debugger to check out your programs without long compile and link passes. Source programs that you develop with QC can be compiled with any of the popular C compilers for the PC. QC's standard library is compatible with those of most compilers. The source code for the library is included, and you can add your own library functions.

QC is affordable: only $60

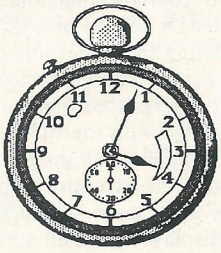
Compare QC to interpreters selling for $99, $150, $249, $300, and $495. Compare QC's full 30-day money-back guarantee. If you are not convinced, send for our $10 demonstration diskette (applied to your purchase) and watch QC in action. We accept MasterCard and Visa.

CST
CST
CST

C SOFTWARE TOOLSET
2983 Newfound Harbor Drive
Merritt Island, FL 32952
(305) 453-0257

CIRCLE 83 ON READER SERVICE CARD

Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!

SRMS™
Software Revision Management System

- Full Featured Revision Control System
- All versions stored in a single ASCII file
- Support for all languages
- Lets you to use your current compilers & editors
- Handles big programming projects easily
- Merge facility to consolidate different development paths
- Versions can contain unlimited comments to aid in the documentation process
- DOS directory and Wildcard Support
- Typeset Documentation with Tutorial

SRMS Version 2.0.....$125

QMAKE™
Intelligent Program Generation Utility

- Controls the building of software systems in a fast efficient manner
- Relieves the developer of the details involved in building software systems
- Analyses complex dependencies in seconds
- Works with most compilers and assemblers
- Full Macro support
- Support for "built-in" or default rules
- UNIX makefile compatability
- Full typeset documentation

QMAKE Version 1.1.....$99
QMAKE + SRMS .....$ 199

TXT TOOLS

NEW! - Qsearch (Like Unix Grep)
- QDIFF (Windowing Difference Utility)
- QSE (Quilt Stream Editor) NEW!

Txt Tools.....$ 85

QUILT COMPUTING
7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

MasterCard VISA

Volume Discounts and Dealer Inquiries Welcome

CIRCLE 84 ON READER SERVICE CARD

and the ability to overlay and remove window panels. Techniques supported by SCREENIO include:

- Positioning window panels at run time
- End-user customization of application colors
- Changeable field attributes, including color, field protection, and cursor positioning
- Control of all fields for editing or file I/O.

An additional SCREENIO feature is the ability to download a CICS (Customer Information Control Systems), BMS (Basic Mapping Support) mapset in ASCII mode and convert it into panels. Once the mapset is converted, you can edit panels and use them to build a BMS mapset for uploading to a mainframe compiler. This feature allows you to download CICS applications for development and testing on a PC, as well as develop CICS applications exclusively on a PC and then upload the finished product to a mainframe.

SCREENIO will add approximately 40K to the size of your application. SCREENIO itself uses approximately 23K of memory, with each panel adding from 1K to 3K of data to your program's *DATA DIVISION*. The additional space required is not prohibitive when you consider that you won't need additional run-time programs—all code is linked to an application. If you elect to use the run-time library, each panel will occupy 316 bytes of *DATA DIVISION* space, plus the storage required for your data fields.

There are no run-time charges or royalty fees on the generated code you produce. While you can run off a floppy drive, a hard disk is recommended. SCREENIO is compatible with DOS v.2.0 and 2.1, but DOS v.3.0 or later is recommended. The SCREENIO subroutine is strictly DOS and BIOS compatible in that no direct hardware calls are utilized, with the exceptions of the memory-mapped video used in SCREENIO's default mode of operation and direct control of the PC speaker.

With SCREENIO, you can generate I/O screens that rival those of Pascal and C programs. The visual effects available to the COBOL programmer are truly remarkable.

Flexus International Screenplay Screen Management System

Screenplay Screen Management System by Flexus International is the least expensive of the products reviewed, but its full-functioned screen management system provides an easy-to-use, powerful system.

The first thing that struck me was the time and effort put into the demo disk. The demo programs not only show all of Screenplay's features but are also an excellent, quick tutorial on using the prod-

uct. Going through the demo several times and using it as a refresher helped me master the program in a very short time.

Screenplay has three major programs for panel definition, management, and control. As you may have guessed, the panel definition facility creates panels while panel management allows operations like copy, add, delete, merge, and such. The panel control facility is the run-time module that links with an application program.

A panel can be any size, from 1 row by 1 column to 25 rows by 80 columns. Panels can be defined to reside at a particular point on screen and then moved at execution time if desired. Also available are full cursor-key control and the ability to program any key or key combination to perform cursor movement, tabbing, backtabbing, insert, or delete functions.

PC graphic characters can be used for visual effects, either in drawing boxes or creating special screen designs. Screenplay offers full control over all attributes, including foreground and background colors and blinking for any character.

One of Screenplay's most impressive features is the ability to design panels and then sit down with a user and run through a prototyping session without having to write any code. The user can enter data into the fields and you'll have valuable feedback before any code is written. I can't tell you how many times I wished I could do this, knowing full well that a user might not like how I designed a panel or the sequencing of the data fields. Long before an application is finished, users can show others how screens will appear and training documents can be produced.

One of Screenplay's greatest strengths is the dynamic control of panel features. With the panel control program you can:

- Dynamically retrieve panel files at run time
- Overlay pop-up panels on displayed panels
- Clear the area occupied by a panel
- Change field attributes including color, blinking, and intensity
- Specify cursor location anywhere
- Change field types
- Alter the location of a panel
- Change control-key assignment
- Change individual characters within a field.

While all panels are indeed maintained in a panel library, I was annoyed to find that you must know how many panels an application has before linking the correct buffer routine in with your application. One obvious solution would be to link the maximum buffer space with all applications.

You Go To A Lot of Trouble To Write Good Applications And Software for Users....

The last thing you want is for the user to have trouble installing your software.....or to install it incorrectly. This just leads to phone calls from the user, wasting your time and money, as well as his.

Why use clumsy .BAT files (or worse) for your software installation procedures when you can get complete control with EZ-INSTALL?!?

EZ-INSTALL allows you to quickly write an installation program that can check the system configuration, including available disk space, memory, file sizes, file contents, time stamps, disk drive availability, DOS versions, directories, etc.

And, with EZ-INSTALL you can easily prompt the user for information and check his response. Plus, EZ-INSTALL has capabilities that give you *complete control* of the installation process---IF, DO, symbolic variables, menus, etc.

If you're distributing your applications or software products to users, you *need* EZ-INSTALL....No compiling, no copy protection, no royalty or license fees.

...only **$69**

For IBM PCs and compatibles
Texas residents add 7.25% sales tax
Shipping $3.50 (US) $10.00 (outside US)
Mastercard, Visa Accepted

To order or receive more information:

(214) 490-0835

The Software Factory
INCORPORATED

15301 Dallas Parkway
Suite 750 LB 44
Dallas, TX 75248

IBM is a registered trademark of International Business Machines Corp.

CIRCLE 85 ON READER SERVICE CARD

TMS™

THE ULTIMATE IN SCREEN GENERATION SCIENCE

Save time by drawing any screen in minutes!
No code to write at all!

FEATURES:

- Built in Screen Compiler (converts Screen Images to Object Modules without Source Code)
- Mainframe style Transaction Processor
- 3270 Keyboard Features
- End User control over Screen Colors
- Realtime Full Screen Editor

MANY MORE UNIQUE FEATURES!

REQUIREMENTS:

- IBM PC, XT, AT, 3270 PC, or 100% compatibles
- Lattice "C" 3.x or Microsoft "C" 4.0
- 256 K

Place your order now!

Price **$195.00**

Tel. **(914) 245-8392**

For more information write to:

WOLF PAK RESEARCH, INC.
90 Upland Rd.
Yorktown Hts., NY 10598

WOLF PAK RESEARCH, INC.

CIRCLE 86 ON READER SERVICE CARD

Systems programming under OS/2 will require a language with concurrency, black-box modularity, procedure variables, and *Pascal-like parameter passing.* PMI specializes in the only language that has them all:

MODULA-2

Our products include:

★ **Repertoire™**: the largest and most successful collection of low-level tools and integrated high-level subsystems for M2. Includes full DBMS with variable-length, keyed records; design/display system for windows, forms, help screens, menus, etc.; expression evaluation and natural-language analysis tools; window-oriented text editor; extensive DOS & BIOS access; thoroughly indexed, 320-page manual; full Modula-2 source code (over 600K) . . $89

★ **ModBase**: an alternate DBMS fully compatible with Ashton-Tate's dBase III; create and access dBase III files from Modula-2, and vice-versa; B+ Tree Indexing system allows files with billions of records.
With full source code $89
Object-code only $49

★ **EXE2LNK**: converts OBJ files created by MASM into Modula-2/86 LNK files
. . . . . $49

★ **Macro2**: a macro preprocessor for Modula-2; provides inline expansion of functions, include files, conditional compilation, etc. With full source code . . $89
Object-code only $49

Call for complete catalog. Full documentation for all products available free of charge.

PMI
VISA/MC
AMEX/COD/PO
(503) 777-8844
BIX: pmi
Compuserve: 74706, 262
The leading supplier of Modula-2 *software components.*
4536 S.E. 50th
Portland, OR 97206
Call for free demo & documentation disk!

CIRCLE 93 ON READER SERVICE CARD

Screenplay is the only screen manager that doesn't produce source code for use directly by a program. Screenplay is a called program that links to your application language or accessed via an interrupt. If your language does not provide a direct link feature, Flexus provides an option to link Screenplay through a dynamic load procedure.

Screenplay's system requirements are similar to FORMIX's. It will run on an IBM PC or compatible with 256K RAM, DOS 2.0 or later, and one floppy drive. Again, while a hard disk is not required, it's recommended.

By the time you read this, Flexus should have released Screenplay v.2. Flexus has stated that it will be priced at $395 for the COBOL version and, notably, will be able to produce COBOL source code. Other promised enhancements over Screenplay v.1 include the generation of COBOL copy books, automatic box drawing from specifications, and saving overlaid panels for restoration. Unfortunately, this version was not available in time to be reviewed.

The COBOL king

The best way to evaluate these products is look at their demos on your own system. Like COBOL compilers, each is targeted for a specific market segment. SCREENIO is probably best for COBOL programmers using Realia's COBOL in a mainframe environment. For those involved in CICS application development, the ability to import and export BMS map sets will be a real time-saver. SCREENIO is also suitable for developers who use Realia to develop products for the open market.

FORMIX offers the full power of a screen management system at the cost of a lengthy learning curve. FORMIX is above the rest in both price and performance. This package is suited to companies that have a COBOL development group and can afford the cost of the package and the maintenance agreements needed to gain full usage from the product.

On the other hand, Screenplay v.1.0 provides the everyday COBOL programmer with all the features of the high-priced spread at a reasonable cost. Screenplay is somewhat limited but can enhance COBOL programs beyond the abilities of mortal man. Screenplay does not have the advanced features for mainframe-programming support. Although 3270 screen design is available, you should use a different product if that's your main interest.

Chuck Ballinger is an information center analyst in Spokane, Wash., working in the area of microcomputer-to-mainframe communications. His interests include packet radio transmissions of computer data.

Features

Manufacturer and product	Copy protected	Royalty	Generates COBOL source code	Panels stored in libraries	Run-time facility	Fields can be changed at run time	Creates panel documentation	Support included	Demo and sample programs	Pages in manual
NORCOM SCREENIO	yes	no	yes	yes[1]	yes[2]	no	yes	yes[3]	yes	275
Flexus International Screenplay	no	no	no	yes[1]	yes[2]	yes	yes	no	yes[4]	181
Master Computer Systems FORMIX	no	yes	no	yes	yes	yes	yes	yes	yes	153

1. Panels stored in panel library or converted to COBOL copybooks.
2. Package requires linked program whether COBOL source code generated or not.
3. Support free for first year.
4. Demo program not included.

Table 1.

WINDOWS—MENUS—DATA ENTRY—SCREENS

HI-SCREEN XL™

Application Developers! **HI-SCREEN** *XL*™, the newest version of HIGH SCREEN, is a complete and unmatched programming tool for managing your user-interface. **HI-SCREEN** *XL*™ features a revamped and enhanced screen editor, increased speed and performance, along with a new thorough reference and user's manual.

Whatever tool you own: compare, you'll be convinced!

1. **WINDOWS**

HI-SCREEN *XL*™ makes you a master in window management:

- Pop-up windows any size, anywhere.
- Use windows for data entry, menus, on-line help.
- Create context-sensitive help, conditioned by user keystroke sequence, with scrollable text.
- Unlimited number of windows, overlapping up to 26 levels deep. Save/restore area under each window.

1. **WINDOWS:** for on-line help, menus, data entry. Up to 26 levels deep.
2. **MENUS:** pop-up, pull-down or Lotus-style, also for batch files.
3. **DATA ENTRY:** automatic field checking: format, type, range... field by field or full screen. Error handling.
4. **SCREENS:** full-featured screen editor. Language independent. Fast display from RAM.

Royalty free, not copy protected, 30 day money back guarantee, trade up available.

Softway, Inc.
PC/SOFT Product Line
500 Sutter St., Suite 222
San Francisco, CA 94102

(415) 397-4666
HI-SCREEN *XL*™ is $149
(CA res. add tax), S&H USA $5
Visa, M/C welcome

PASCAL, C, dBASE, BASIC, COBOL, FORTRAN, ASSEMBLER, etc.

CIRCLE 88 ON READER SERVICE CARD

C to dBASE

the dBx™ Translator

- C from dBASE II, III, III+ programs
- Move to UNIX, XENIX, QNX, MAC, AMIGA
- Faster, more reliable IBM PC programs
- Supports multi-user and network
- Run your code on any standard C system
- Know dBASE? Learn C easily.
- Priced from $350; available from distributors
- Includes full screen handler library
- Supports a choice of C database managers

from Desktop Ai

1720 Post Road East, Westport, CT 06880
Telephone: **203-255-3400** • Telex: **6502972226MCI**
MCIMAIL-DESKTOPAI

dBASE is a trademark of Ashton-Tate dBx is a trademark of Desktop Ai

CIRCLE 89 ON READER SERVICE CARD

KOMPUTERWERK

Your BASIC Resource

Specializing in Libraries & Utilities for Compiled BASIC

QuickBASIC	$79.00
FINALLY! A Subroutine Library for Compiled BASIC	$99.00
FINALLY! Modules (Window and Menu Programming Modules)	$99.00
GDPEDIT Graphics Editor	$99.00

Include $4.00 shipping and handling for each item

Call or write for our FREE catalog of products for Compiled BASIC programmers

Information (412) 782-0384 9:00 AM–5:00 PM EST

Orders 1-800-423-3400 9:00 AM–8:00 PM EST

KOMPUTERWERK
851 Parkview Blvd.
Pittsburgh, PA 15215
(412) 782-0384

Powerful tools at reasonable prices.

CIRCLE 90 ON READER SERVICE CARD

Programmers:

Turbo-charge your productivity with PL/PC

PL/PC is a new programming language based in large part on APL (A Programming Language) with Modula-2 control structures. It offers an integrated interactive programming environment for the rapid implementation of applications.

Structured programming is supported with Modula-2 control structures, block structured declaration of subroutines and automatic paragraphing of subroutines. Multi-dimensional arrays are easily manipulated with the large set of PL/PC array operators. Fundamental data types are extended to include complex numbers and strings. A full-featured full screen text editor is included, the editor will automatically position the cursor at the point in the source code where the last compile-time or run-time error was detected. Data are edited with a spreadsheet like data editor. English keywords are used instead of APL symbols, eliminating the requirements for special keyboard, character generator and printer. DOS files can be structured to be manipulated as a single data item of any dimension or data type. Graphic applications are supported with routines to draw lines, points, polygons, circles, conic sections and manipulation of data to/from screen. Debugging facilities include tracing, stopping, single-stepping, timing and profiling.

A demonstration version is available for US$16. The demo version comes with a reference manual and it has a limitation of six global variables. The standard version is priced at US$89 and the 8087 version at US$159. All prices include airmail postage and handling.

PL/PC requires an IBM PC or compatibles with at least 360K of memory and DOS 2.11 or higher.

Creative Computer Software
117 York St., Sydney, NSW 2000, Australia.
Phone: (02) 261 1611 Fax: (02) 264 7161

CIRCLE 91 ON READER SERVICE CARD

Curses!

By Mark Davidson

In recent years, the UNIX operating system has had a big impact on PC programming tools. Many programmers want the power of UNIX tools but hesitate to run UNIX (if it's even possible). The IBM PC and compatible market reflects this fact; PC/MS-DOS has begun to resemble UNIX. Unfortunately, that operating system's tools are minimal.

In response, many developers have released utilities based on the UNIX tools. These tools have appeared in the public domain (versions of nroff, ls, sccs, vi, make, and grep) and as commercial products (MKS Toolkit, PC/VI, and such).

One area generally left to compiler writers has been UNIX-like library functions for program development. Admittedly, today's compiler libraries supply a vast number of functions for the programmer (generally based on the UNIX System V libraries). However, one of UNIX's more powerful libraries, curses, is just now making its way into the market.

When UNIX was first developed many years ago, only line-oriented, teletype terminals were available. In time, these were followed by a variety of full-screen, ASCII terminals, each with its own features (or lack thereof).

At that time, most of the UNIX tools had been written for the teletypelike terminals and did not take advantage of the full-screen terminals' capabilities. If you wrote a program to take advantage of a terminal's features, it ran on that terminal only. To get a program to run on another terminal, the software had to be changed, and some of the terminals were just strange enough to make this difficult.

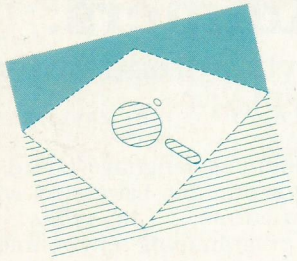
From the University of California at Berkeley came a solution, the release of two tools for UNIX: termcap and curses. Termcap was a library of terminal capabilities that contained definitions for terminal types and what they could do. (It used a series of two-character codes for different capabilities.)

Curses was a library of screen management routines for controlling multiple windows and screen buffers. The programmer decided what the screen should look like and, with a function call, curses would decide the best way to update the screen.

With curses, a programmer could write a full-screen application that would work with almost any terminal defined in the termcap data base. Of course, before using the application a user had to set an environment variable to describe the terminal type during log-on. When a program using the curses library was run, curses would check the terminal type. If the terminal was not "dumb," it would also check the termcap data base for the terminal's capabilities. Curses would then know how to do such things as position the cursor, highlight, and clear the screen.

All curses functions work with a structure of type *WINDOW*. This type is defined in curses.h, which must be included in any program using curses. In working

The 1st annual conference *exclusively* for software developers!



SOFTWARE DEVELOPMENT '88

SAN FRANCISCO, FEBRUARY 17–19, 1988

The publishers of COMPUTER LANGUAGE, AI EXPERT and UNIX REVIEW have joined together to expand on the successful COMPUTER LANGUAGE C Seminar. We plan to create the most practical and useful programming seminar and trade show possible. Become involved in this event from the start by filling out this coupon today.

Name _____

Company _____

Address _____

City, State, Zip _____

Phone _____

Send to: Beatrice Blatteis, *Software Development '88,* Miller Freeman Publications, 500 Howard Street, San Francisco, CA 94105.

☐ **Please put me on your mailing list** to receive further information on attending the seminar and trade show as it becomes available.

☐ **I may be interested in exhibiting at the trade show.** Please send me further information as it becomes available.

☐ **I would be very interested in these seminar topics:**

_____

_____

☐ **I suggest the following speakers:**

_____

_____

☐ **I would like to present a paper about:**

_____

with curses, you can define your own *WINDOW*s, add characters to a window, specify the border of a window, and much more.

System III curses had a total of 84 functions (some of which are actually macros). Eighty-four may sound like a lot to learn, but many are just variations. For example, you would use *deleteln* to delete a line from a standard window and *wdeleteln* to delete a line in a specific window.

What is a standard window? Curses allocates a default window, *stdscr,* when a program starts up. *Stdscr* is curses' way of addressing the screen itself. You don't have to allocate *WINDOW*s unless you need them. You can use curses to do such things as position the cursor and delete lines on the screen.

In actuality, all calls that don't reference a specific *WINDOW* structure are macros that get changed to *w*-prefixed calls. For example, if you want to delete a line on the screen at the current cursor position, you would use *deleteln()*. The C preprocessor translates this using the macros defined in curses.h to *wdeleteln (stdscr)*.

Practically speaking, you only have to memorize half the function names; the other functions are just variations with a prefix attached. Prefixes come in three varieties: *w, mv,* and *mvw*. As shown for *deleteln,* some calls take a *w* for action on a specific window buffer. *mv* and *mvw* are attached to some functions to specify a cursor position and an action in the same call. For example, to move to row 3, column 5 and delete a character, you could type:

```
move(3, 5); delch();
```

The *mv* prefix shortens this to:

```
mvdelch(3, 5);
```

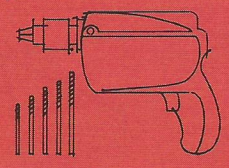
The *mvw* prefix allows you to specify a cursor move, an action, and a specific window buffer, all in one call.

Curses does not actually update the screen every time you make a call to one of its functions. You issue as many (or as few) calls as you want and then call *refresh()*. Curses then updates the specified buffer on screen as fast as possible. If the buffer being updated is not the size of the full screen, any area outside the buffer is not touched. By default, *stdscr* is the entire screen. With this design, you can allocate small windows that take up only a portion of the screen and your program need only handle small portions of the screen at a time.

With all this said, why use curses? Curses offers many features to a programmer, and many programs use curses. Also, some PC programs are being developed under UNIX and then ported to the PC. A PC-based curses library speeds up the port. Another reason to use curses might occur when a UNIX programmer wants to program in C on his or her PC at home and doesn't want to learn another library to do screen management. Finally, a PC programmer might just want to learn about curses.

I'll look at two PC versions of curses in this review: Lattice Curses Library and Aspen Scientific Curses Window Development Library. I had the opportunity to use both the Lattice and Microsoft version of each product and will try to point out differences between the two when appropriate.

In assessing the products, compatibility with UNIX curses is an important feature. Otherwise, a library is useless for UNIX programmers working on a PC. A library should also take advantage of the memory-mapped video of PC display cards to provide fast screen updating without flicker or snow. I also considered how PC-specific a library is, whether it runs on compatibles, if color is supported, and last, but not least, if you need to be a curses wizard to use the product.



POWER TOOLS for SYSTEM BUILDERS™

Trust TSF to provide the best value in professional development tools

* Technical advice **and support** by programmers
* Honest and equitable business practices
* Prompt, flexible service to meet your needs

Bytel **Genifer** dBase III application generator **$250**, *Wordtech* **dBxl** dBase III work-alike **$110**, *Borland* **Turbo Basic $68**, *MicroHelp* **Mach2 for Turbo Basic** fast i/o and error trapping **$55**, **Periscope I** debugger with protected memory card and NMI switch **$279**, *Turbo Power* **Tdebug+** symbolic debugger for Turbo Pascal **$48**, *Xerox* **Ventura Publisher** full featured and efficient -- works with XTs **$599** ... MANY MORE ... CALL OR WRITE FOR FREE CATALOG

Free Catalog!

The Software Family
649 Mission Street
San Francisco, CA 94105

(1)800-443-7176

In California: (415)583-4166

__ Please send me a free catalog, I'm most interested in: __ Basic __ C __ Pascal __ Turbo Pascal __ dBase __ Other Languages __ Publishing

__ Please send the following products:

Name: _____

Address: _____

City/State/Zip: _____

Credit Card: _____

Exp: _____

Mail to: TSF
649 Mission Street
San Francisco, CA 94105

CIRCLE 95 ON READER SERVICE CARD

Aspen Scientific Curses Window Development Library

Aspen Scientific's curses library has much to offer the programmer interested in curses on the PC. Not only does it support several different C compilers, but it includes fairly good documentation and two interesting demo programs.

Aspen Scientific's library is based on System V curses. It supports almost all of that version's functions except for processor timing and delay functions. I tested the library in three ways: by going through the tutorial, compiling and altering the demo programs, and porting an older System III curses demo program I had used under PC/IX.

The Aspen library comes on one 360K disk with versions of the library for each memory model. Also included are two demo programs (pop and yavs), a version of the UNIX make utility, and extended curses functions with source code. A make file (for use with the make utility) includes the necessary commands to recompile the demo programs. My review copy came with a second disk for use with older IBM XTs' BIOS.

The Aspen documentation is well done, with 102 pages that are mostly a reference to the library's functions. The introduction has short sections on porting UNIX programs to the PC, porting PC programs to UNIX, why you should use curses, terminal types supported (more on this later), and how to program with curses. A short tutorial, "How to Write a Curses Program," gives a brief introduction on what a curses program looks like and encourages the user to explore the demo programs. A more in-depth tutorial would be nice, but the demo programs are straightforward.

The rest of the manual is devoted to the curses functions. They're listed in typical UNIX fashion with each function's description including sections on usage, arguments, return codes, references, and examples. No index is provided, but a quick reference section for all the functions includes page numbers for each function's complete description. The manual is printed on 8-½-x-11-inch paper and uses multiple fonts that make the manual easy to read.

Aspen's demo programs are worth mentioning. The first one, pop, shows how to implement pop-up windows. The second one, yavs (Yet Another Visual Shell), implements a DOS front-end for listing directories, viewing files, and more. Both programs are designed to show what's possible with curses. Yavs makes heavy use of color, pop-up windows, and function keys.

One thing I noticed about the demo programs is that the boxes look complete (the corners are drawn correctly). Once you call the box function with the box-drawing characters, Aspen's box function will pick the corner characters from the IBM character set. To make this even easier, Aspen includes box.h, which contains definitions for all the box characters. Aspen provides several ways to update the screen; you can even change the way Aspen updates the screen without recompiling your program.

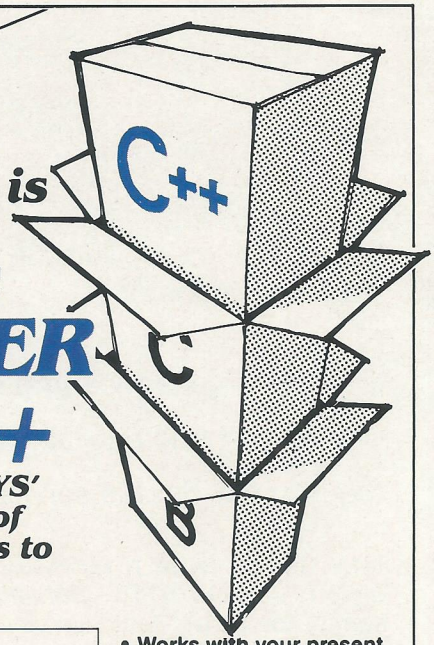
When *initscr()* is called, the Aspen library looks for the environment variable *TERM*. *TERM* can have one of four settings: *ibm-pc* (for color or monochrome memory-mapped output), *bios-pc* (for color or monochrome IBM BIOS output), *ansi-pc* (for monochrome ANSI.SYS-compatible output), and *c_ansi-pc* (for color ANSI.SYS-compatible output). Thus a program can run on just about any PC or compatible and can be tailored to a user's environment without you having to recompile the program or sell different versions for different displays. If the *TERM* environment variable doesn't exist, the default value is *ibm-pc*.

Aspen provides several extension functions and their source code. These functions are not standard UNIX functions but ease some of the internal inconsisten-

Now for VAX C & Sun C

The Wrapping is off the Latest Evolution of C

DESIGNER C++



Designer C++ is OASYS' full implementation of AT&T's enhancements to the C language

FEATURES:
- ► Optional strong type checking
- ► Data abstraction
- ► Optional guaranteed initialization of data structures
- ► Overloading of function names and operators
- ► Dynamic typing (virtual functions)
- ► Optional user-defined implicit type conversion

- Works with your present C Compiler
- Functions as a Pre-processor/Translator: handles regular C code with no changes
- Type-checking and other features are optional — you can turn them off
- Already thousands of users at commercial sites
- Complete documentation: C++, A User's Guide by Bjarne Stroustrup of AT&T (Addison-Wesley, 1986)

The only commercially-available C++ customized to operate on PC's, micros, minis, and mainframes with popular C compilers, including:

VAX C
SUN C
ULTRIX C
GREEN HILLS
LATTICE*
MICROSOFT*

*Lattice and Microsoft versions of Designer C++ are known as Advantage C++

We Specialize in: Cross/Native Compilers: C, Pascal, FORTRAN, Ada, LISP — Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Profilers — QA Tools — Design Tools — Comm. Tools, — OS Kernels — Editors — VAX & PC Attached Processors and more
We Support: 680xx, 80x86, 320xx, 68xx, 80xx; Clipper, and dozens more

A DIVISION OF XEL Oasys

60 Aberdeen Ave., Cambridge, MA 02138 (617) 491-4180

Designer C++ is a joint trademark of XEL, Inc. and Glockenspiel, Ltd of Dublin. Ada is a trademark of the U.S. Government (AJPO). Advantage C++ is a trademark of Lifeboat Associates, Inc. Other trademarks are acknowledged to DEC, Lattice, Microsoft & Sun Microsystems, Inc

CIRCLE 61 ON READER SERVICE CARD

cies between Aspen's package and standard curses. Functions are provided for such things as finding out if a window is scrollable, the current *TERM* value, the status of the shift keys, and the limits on a window's *x* and *y* coordinates. Function keys and other extended keys are supported through the use of a *keypad()* function. Aspen *#define*'s values for your program so that it can easily access these keys. Color is supported through attribute values. These constants are defined for you in the curses header files, which are *OR*ed together in a call to *wattrset()* to turn colors on or off (a much easier method than Lattice's external variables).

I compiled and ran the System III demo program with no changes whatsoever using the Aspen library and Microsoft C. Screen updating was fast and clean. Due to a problem with the Lattice compiler, I could not run this program with Lattice C and the Lattice version of the Aspen library.

Aspen Scientific's curses library is a fine PC version of System V curses. The library is an almost complete implementation of System V curses, has good documentation, and is available for many compilers. The Aspen package is less expensive than Lattice, and its greater flexibility makes it an attractive package for developers.

Lattice Curses Library

Lattice markets several add-on libraries for the Lattice C compiler that are targeted at developers. After all, who should know the compiler better? Among these products is Lattice Curses Library, an adequate implementation of curses that suffers from several annoying problems.

Lattice Curses Library v.1.05 is based on System III curses and thus is not as complete as an implementation of System V curses. As with the Aspen Scientific library, I tested Lattice curses in several ways: by going through the tutorial, by compiling and altering the demo programs, and by porting an older System III curses demo program I had used under PC/IX.

The Lattice product comes with one 360K disk that contains two demo programs and versions of the curses library for each memory model. Also included are several batch files, source code for two curses functions (*wrefresh* and *initscr*), and curses.h (which you include in any program that uses the library).

The batch files are for loading the product into Lattice C's subdirectories, linking your programs with the curses and standard Lattice library, and removing the curses library (if desired). Unfortunately, the install batch file assumes all memory models of the Lattice compiler

HELP

is at hand

HELP/Control™ — an on-line help system for the IBM-PC. **HELP/Control** includes **HELP/Runtime**, **HELP/Popup** and our help screen compiler.

With **HELP/Runtime**, a few simple subroutine calls add context sensitive on-line help to your application. **HELP/Runtime** includes tested interfaces for C (Microsoft and Lattice), Pascal (Microsoft and Turbo), IBM BASIC (Interpreter and Compiler), Microsoft FORTRAN, COBOL (IBM and Realia) and assembler.

Use our concise screen definition language to build your help files. You define the bold captions on your help screens and specify the links to other screens. If you have existing documentation files, we supply a program which automatically marks them up to get on-line quickly. You can put an entire user or reference manual on-line, completely accessible to the user at all times.

HELP/Control also includes **HELP/Popup**, which provides memory resident access to your custom help screens. Use it to document dBase and 123 applications. **HELP/Popup** uses the same help files as **HELP/Runtime** and operates the same from the user's point of view, allowing you to provide a consistent on-line environment across diverse applications.

The complete package (software, on-line manual, printed manual, and demo programs) costs $125.00 and includes a royalty-free license to add **HELP/Runtime** to your applications and a license to make 25 copies of **HELP/Popup**. A demonstration diskette, including the on-line manual, costs $15.00. To order, or for more information (including dealer, multiple-copy and site-license pricing) call MDS at 207/772-5436. We accept MasterCard and VISA.

New options with release 1.20:

- **Use Dan Bricklin's Demo Program to build your help system. Our translator turns it into a HELP/Control source file.**
- **Include graphics in your help screens.**
- **Mouse support.**

MDS

MDS, INC., P.O. BOX 1237, PORTLAND, MAINE 04104

CIRCLE 96 ON READER SERVICE CARD

Personalize your computing environment.

The MKS Toolkit now contains the Korn shell command interpreter.

The MKS version of Bell Labs' Korn shell has this and more:

- the full power of the UNIX System V.2 Bourne shell
- the most requested features of Berkeley's C shell
- the full-UNIX utility of executable shell files
- command aliases
- interactive command-line facilities
- previous command history and editing
- a powerful programming language
- shell variable expansion
- arithmetic evaluation

All this has been fine-tuned to create the optimum environment under DOS. The Korn shell is just one of over 100 commands — fully compatible with UNIX System V.2 — now contained in the MKS Toolkit, including the following:

awk	cat	chmod	cmp	cp	cpio	ctags	cut	date
dd	df	diff	du	echo	ed	egrep	ex	fgrep
file	find	head	help	join	lc	ls	more	mv
nm	od	paste	pg	prof	rm	sed	size	sort
split	strings	tail	time	touch	tr	uniq	vi	wc

and much, much more . . .

These programs run from the shell or command.com under DOS on machines such as the IBM PC, XT, and AT, the AT&T 6300, and most PC compatibles. Full documentation is included. Phone support is available 9-6 EST. Not copy protected.

Everything for only $139.

Mortice Kern Systems Inc.

43 Bridgeport Road East, Waterloo, Ontario, Canada N2J 2J4

For information or ordering call collect: **(519) 884-2251**

Prices quoted in U.S. funds. MasterCard and VISA orders accepted. OEM and dealer inquiries invited. UNIX is a trademark of Bell Labs. MS-DOS is a trademark of Microsoft Corp.

CIRCLE 97 ON READER SERVICE CARD

MULTITASKING

Introducing

MultiDos Plus

The new multitasking software for the IBM-PC.

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in any language.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-128 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Monochrome/CGA display adaptors or equivalent cards only. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

ONLY **$49.95**

Outside USA add $5.00 shipping and handling.

Visa and Mastercard orders call toll-free: 1-800-367-6707. In Mass call 617-651-0091, or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760

MA orders add 5% sales tax. Write for source code and quantity price.

CIRCLE 92 ON READER SERVICE CARD

are installed. If they're not, it would probably be better to copy the library yourself. The install batch files could conceivably copy a memory-model library to the wrong place. The Microsoft version of the library does not include installation batch files.

The manual is probably the weakest part of the package. Topping out at 30 pages, it's printed in a small, light font that's hard to read. All the needed information is there, but the manual has no index or table of contents. All that's included is a list of topics in the front; I assume Lattice thinks you don't need either of the aforementioned with only 30 pages to look through.

The manual consists of a small introduction, product description, short tutorial, function reference section, list of functions provided for compatibility only, and quick reference section. The functions are grouped by category, and each function is shown with the arguments required and type returned.

Following this information is a short description of what the function does. "Short" is the key word here. Most of the descriptions are very short; only a few of the more complicated functions have longer descriptions. Needless to say, the manual needs improvement.

The Lattice library performs fairly well. When a program calls *refresh()*, the current screen buffer is moved to the screen using direct video memory access. Lattice Curses waits during the retrace interval so no snow appears on screen during updating.

Lattice Curses offers two other methods of updating: the screen can be updated without waiting for retrace intervals, as for a Compaq display or an EGA, or the display can be turned off while the update is being performed (if you don't mind the display blinking as it does when a *dir* is scrolled).

Unfortunately, changing the method of updating can't be done on the fly. You'll have to change *refresh()* (source code is provided), recompile it, and then relink your programs. With Lattice's update method, you'll need a very IBM-compatible display.

The demo programs are minimal and don't do much to show off the product's capabilities. Mostly they just display several windows, show window movement, and demonstrate how to handle keyboard input. No color is used (although it is supported) and the window borders are drawn with characters like *!*, ***, and *$*. No box characters from the IBM PC character set are used. You can use them in your programs, but the corners won't turn out the way you might expect.

As mentioned, color is supported. To use color or any special display attribute, you place the attribute value in a global variable called *s_attr* and then call *standout()*. Any character I/O to the screen will then use this attribute. The default value for *s_attr* is 0x70, which displays characters in reverse video. (This is also the default under UNIX, but note that *s_attr* is not available under UNIX.) This method is a logical way to handle color but could be awkward if you wanted to use many different attributes for your displays. To display characters normally again, you call *standend()*.

Actually, three external variables can be used to control attributes: *s_attr*, *n_attr*, and *f_attr*. *s_attr* is for standout displays, *n_attr* is for normal displays, and *f_attr* is the filler attribute. These variables are set to the attributes in effect when your program begins execution. *f_attr* and *n_attr* are not mentioned in the manual but are described in a READ.ME file.

This brings me to a point about the curses library function *box()*. Under UNIX, a call to *box()* might look like: *box(stdscr, '', '_')*. This call is interpreted as draw a box around *stdscr* using as the vertical box character and _ as the horizontal box character.

On an ASCII terminal under UNIX, all this makes perfect sense. Most terminals don't support box characters like the ones found in the PC's character set, and you therefore only have to specify two border characters. With the characters available on the PC, boxes look much nicer if you use the appropriate corner characters. Under Lattice Curses, you can't do that. Seeing the Lattice demo boxes drawn with border characters like * didn't impress me.

I was unable to get my System III demo program to work with the Lattice C compiler and Lattice Curses. I did run the program using Microsoft C and the Microsoft version of Lattice Curses, but the screen update was slower than Aspen Scientific's for the same program. The Lattice library supports all of the System III curses functions, but not all are implemented. Several of the functions are

Curses products and manufacturers

Aspen Scientific Curses
Window Development Package
Package—$119
Source code—$289
Aspen Scientific
P.O. Box 72
Wheat Ridge, Colo. 80034-0072
(303) 423-8088

Lattice Curses Library
Package—$125
Package with source code—$250
Lattice Inc.
P.O. Box 3072
Glen Ellyn, Ill. 60138
(800) 533-3577
(312) 858-7950 in Ill.

UNIX-specific and meant to deal with terminals. When you compile a program that uses these functions, the supplied macros expand to nothing, which shouldn't affect your programs. If you compile a program using the functions, you'll get a "statement has no effect" warning. If you're porting a program, this design is helpful since you don't have to go looking for the needed functions.

One point worth noting with the Microsoft version of Lattice Curses: my review copy would not work with the cl front end because cl invokes the linker (by default) with the */NOI* option. One of the functions in the Lattice Curses library, *vidpoke()*, was compiled and assembled with the name *VIDPOKE* (all uppercase) and thus was not found when I tried to compile one of the demo programs that used it. The msc front end does not have this problem because msc does not invoke the linker automatically. Leaving off the */NOI* option allowed the program to link.

Lattice Curses is a no-frills, vanilla version of UNIX System III curses. It uses direct video memory access for fast updating of the screen but could give problems on some compatibles. Color is supported but can be awkward to use.

UNIX programmers with curses experience will have no problems using this library. However, because of the minimal tutorial and small manual, a novice will have a long learning curve ahead. You can purchase the source from Lattice if you want to see how curses works from the inside.

Finally, price is a factor. The retail price is $125. This still seems high when products like Aspen Scientific's are based on a more recent release of curses and offer better documentation for only $119 retail.

Mark Davidson is a mainframe applications programmer in Chattanooga, Tenn., interested in UNIX and DOS programming utilities.

The Productivity Tree

System requirements: IBM PC/XT/AT or compatible, 128K RAM, DOS, one floppy drive, monochrome or color graphics
Price: BASIC version, $99.95; C version, $149.95
Available from: Programming Technologies Inc., P.O. Box 370, Pennington, N.J. 08534, (609) 737-0505
Support: Phone support tree to registered users

Does it seem reasonable to be forced to develop source code modules in a programming environment that allows only 20 lines of code per module? Such a limitation might strike you as extremely confining and not conducive to programming creativity.

Yet this is precisely the limitation imposed by Programming Technologies' The Productivity Tree. Once you have reshaped your thinking about program development, you may agree that a 20-line limit has much in its favor.

The Productivity Tree (PT) is a programming system available for BASIC and C on IBM and compatible microcomputers. It was originally developed for COBOL and PL/I on mainframe computers. The philosophy underlying PT is tree-structured thinking. Tree-structured thinking is normal to most of us. As we work through complex problems, we move along limbs, branches, and smaller offshoots. One idea suggests another, related idea, which in turn suggests another, and so on. With PT, you can create modules that follow the flow of your logic.

PT modules are merged into a standard program format that can be used with BASIC or C compilers and interpreters. The number of modules is limited only by disk space.

The modules are stored in a compressed format and must be edited with the PT editor. This full screen editor has all the features needed for writing and modifying programs. Standard source code can, of course, be edited directly, but this quick-and-dirty approach negates the value of the 20-line modules and their maintainability.

Each module can hold program code, pseudocode, and comments. Since a given module as displayed is always complete, the programmer gets a clear picture of the purpose and operation of each module. More importantly, the 20-line limit forces the programmer to break the program into comfortable and easily maintainable units.

PT provides for three module types: in-line (ordinary program chunks), subroutine, and data. During the source code merge, PT inserts in-line modules wherever they're invoked in the final program. Subroutines are inserted once and called from *GOSUB* statements as needed. Similarly, data modules are inserted once in the appropriate portion of the program.

Why would you want to use module types? For functions referred to only once or functions of only a few lines, an in-line module is most efficient. PT will merge the same in-line module into every program location that references a given module.

Modules are stored in a data base.

Each module is labeled with a number and by type, name, and number of referencing modules. The data base can be accessed from the module data base maintenance menu, or you can use the editor to list the modules and lines that reference a module.

For documentation, a *REMARKS* command allows you to append a full screen of remarks to any module. In BASIC, PT will automatically *REM* this module adjunct during the merge process, though you must explicitly indicate any *REM*s that appear in the module proper.

Does PT make a meaningful difference for practical programming? Will you become a more skillful programmer? Will programming drudgery be diminished? In varying degrees, the answer to all of these questions is yes.

With PT's conceptually simple programming environment, BASIC becomes a set of keywords. Prior notions of the appearance and stylistic limitations of a BASIC program can be forgotten and replaced with PT modular programming using BASIC keywords. Ditto for C.

The point is that PT modules can stand for any logical program function, regardless of the BASIC or C elements used. As long as you use the correct keyword syntax, the PT sequencer will transform your PT program into a proper BASIC or C program.

The 20-line limit does force you to think carefully about a program's structure. What would have been large, clumsy modules in a conventional language must be broken into smaller, more manageable units.

Each module tells its own complete story. Each is self-contained since it accomplishes a single purpose and invokes appropriate ancillary routines.

During program development, you can explore the module tree being generated with the *NEXT* and *QUIT* commands. Tracing a program through PT is considerably easier and more informative than trying to decipher a printout of the program. With each module linked in a logical chain, the program design can be clearly grasped—especially so with the extensive documentation allowed.

The best approach to using PT is to pseudocode each module before writing detailed code. Think of it as flowcharting on the fly. The sequence in which you create PT modules is left to your discretion. Regardless of development order, the program's logic flow will always be top-down.

In addition to the data base editor and program assembler, the PT system includes utilities for module printout and deletion, program-to-program module copying, and module generation from existing BASIC or C programs. Despite requiring some additional work on the part of the programmer, the module generation is impressive.

System requirements are modest, but since the program does access the disk frequently, you'll be happier running it from a hard disk. Everything ran swiftly and smoothly on my 4.77 MHz PC compatible and hard disk. PT seems well protected against user mistakes. However, if you try to load a nonexistent or misnamed module data base, you're unceremoniously dumped back into DOS.

The only significant difference between the BASIC and C versions of PT is the resultant source code. The $50 price difference may be due to the difficulty of developing a C code generator.

The 8½x11-inch, ring-bound manual is extensive (244 pages), clearly written, and provides enough detail to cover virtually any question. Indeed, the explanations of system messages fill 86 pages. A sample program is included in the manual and on the single, non-copy-protected disk.

Before you reap the benefits of The Productivity Tree, you'll probably suffer through some frustration as old habits give way to new programming techniques. If you're thinking of trying The Productivity Tree modular programming system, count on moments of being sorry you ever got started with it. At some point, though, your patience will pay off.

By Henry F. Beechhold

Add Data Security to Your C Programs

The SECURITY LIBRARY™

Add Fast or Thorough Encryption or Compression to Your Programs WITHOUT Royalties

Build the safest and most popular methods of protecting your data directly into your program without paying royalties. Don't worry if you're not sure what to use. The discussion of security methods in the manual (with demos on disk) will help you make a choice based on your application.

With Security Library you can:

- Keep files secure on a multi-user network
- Speed up data transmission and communications through data compression
- Control access based on privileges given to a hard disk or LAN user

"I create custom software for business applications. I'm using the Security Library to encrypt so that certain information will not be readily available. The documentation is excellent. It gives a thorough explanation about how you can secure a file."

— Bruce Philips, Custom Software Design
Virginia Beach, CA

Requires MSDOS 2.0+. Works with Microsoft C and Computer Innovations' C86. Please specify compiler when you order.

You Choose the Security Level

Algorithms provided include The National Bureau of Standards' Data Encryption Standard (DES) and the Vernam and Vegenere ciphers. Encrypt a 10K file in 3 seconds with one method or 50 seconds with another.

Six algorithms are provided, along with password and non-password encryption schemes.

Valuable Extras

The Huffman coding routine can *reduce the size of a file* by 25 to 50%. The routine to change the attribute bytes of a file can make that file invulnerable to casual browsing or accidental deletion. There's even a program to *change every byte of a file to a null character* - not even un-erase programs can recover it then!

Call 800-821-2492 to order Security Library risk-free for only $125. Source & Object is $250.

Solution Systems™
335-L Washington St.,
Norwell, MA 02061
(617) 659-1571

Full refund if not satisfied in 30 days.

CIRCLE 98 ON READER SERVICE CARD

OPT-TECH SORT™

Sort/Merge/Select Program
for the IBM-PC and Compatibles
New Version 3.0 with
Btrieve & dBASE support

- High performance assembly language
- Sort multiple files on multiple keys
- Filesize limited only by disk space
- Select records on up to 10 keys
- Output records can be reformatted
- Supports many filetypes and datatypes
- Can be CALLed or used stand-alone
- $149 — Visa, MC, Amex, COD, or PO
  Quantity discounts available.

To order or to receive more information call or write:

Opt-Tech Data Processing
P.O. Box 678, Zephyr Cove, NV 89448
(702) 588-3737

CIRCLE 99 ON READER SERVICE CARD

Quelo® 68000 Software Development Tools

First release 1983 - MOTOROLA compatible - produces ROMable code, S-records, extended TEK hex, UNIX COFF. Portable SOURCE CODE. Native and cross versions on: ATARI ST, AMIGA, Masscomp, Sun, Apollo, Charles River, VAX VMS and UNIX.

68020 Cross Assembler Package
Supports 68000, 68010, 68020, 68881 and 68851
For CP/M-68K and MS/PC-DOS - $750

68000/68010 Cross Assembler Package
For CP/M-80, -86, -68K and MS/PC-DOS - $595

68000 "C" Cross Compiler
For MS/PC-DOS by Lattice, Inc. - $500

NEW ITEMS

68020 Disassembler
Supports 68000, 68010, 68020, 68881, 68851
For CP/M-68K and MS/PC-DOS - $495/295.
Amiga and Atari ST - $119/79. CRDS UNOS - $995/595

68000/68010 Software Simulator
For MS/PC-DOS by Big Bang Software, Inc. - $285. VAX - $1900

Call **Patrick Adams** *today:*
Site, Corporate, OEM licenses
COD, Visa, MasterCard

Quelo, Inc.
2464 33rd. West, Suite #173
Seattle, WA USA 98199
Phone 206/285-2528
Telex 910-333-8171

TM: Quelo, Quelo, Inc.; MS, Microsoft Corporation; CP/M, Digital Research

CIRCLE 100 ON READER SERVICE CARD

STRUCTURED™
INFORMATION SYSTEMS, INC.

FINALLY . . . ASSEMBLY LANGUAGE CONTROL WITH COMPILER CONVENIENCE!

- ASSEMBLY LEVEL LANGUAGE WITH SYNTAX SIMILAR TO "C"
- HIGH LEVEL CONTROL CONSTRUCTS INCLUDING: IF-ELSE-WHILE-SWITCH-DO-WHILE
- INCREASES PRODUCTIVITY
- FULL MACRO PREPROCESSOR
- GENERATES INTEL OR MICROSOFT ASSEMBLY LANGUAGE
- RUNS UNDER PC/MS DOS 2.x AND 3.x 256K
- AVAILABLE FOR 8086/186, 8051, OR 8096
- $99 CHECK, MONEY ORDER, VISA, MASTERCARD

TO ORDER, CALL OR WRITE

STRUCTURED™
INFORMATION SYSTEMS, INC.
86 BROAD ST., EATONTOWN, NJ 07724
1-201-544-8388
4-6 Weeks Delivery
N.J. Resident Add 6% Sales Tax

CIRCLE 106 ON READER SERVICE CARD

function libraries
disassemblers
compilers
text editors
text filters
communications support
text formatters
interpreters
bulletin boards
co-routines
compiler compilers
window packages
assemblers
games
tutorials
math packages
link editors
languages
cross compilers
pre-processors
function libraries
disassemblers
compilers
text editors

Users' Group Library

A Directory of Public Domain C Source Code

Send $10 for Directory. Write or call for more details on over 100 volumes of Public Domain C Source Code.

The C Users' Group
PO Box 97
McPherson, KS 67460
(316) 241-1065

CIRCLE 102 ON READER SERVICE CARD

VERSION CONTROL SYSTEM

TLIB™ keeps ALL versions of your program in ONE compact library file, *even with hundreds of revisions!*

- **Fastest, most powerful** version control system you can buy. *Nothing else comes close!* TLIB updates libraries faster than some text editors can load and save files.
- *LAN-compatible!* Shared libraries with PC Net, Novell, etc. Check-in/out locking for multi-programmer projects.
- Synchronized control of multiple related source files.
- *Easy to use.* Menu or DOS command line parmeters.
- *Frugal with disk space.* Libraries are more compact than with most other version control systems. And TLIB uses no temporary files, so you can maintain a 300K library on one 360K diskette, with room to spare, even with TLIB itself on the same disk.
- Free copy of Landon Dyer's excellent public domain **MAKE** utility (.EXE, plus source code for DOS & VAX/VMS).
- **Plus:** File compare utility. Virtually unlimited source file size. Date, comments with each version. Configurable user interface, and many configurable options, like: read-only libraries; automatic tab/blank conversion; insertion of revision history comment block in the source file.

PC/MS-DOS 2.x & 3.x ***Just $99.95** + $3 s/h* Visa/MC

BURTON SYSTEMS SOFTWARE
P. O. Box 4156, Cary, NC 27511-4156
(919) 469-3068

CIRCLE 103 ON READER SERVICE CARD

Save Time & Money
With
GeneratoR

"The Structured Program Editor and Source Code Generator that lets you design before you program"

--Saves you 30% or more in coding time....
--Create Documentation as you program....
--Computer aided software engenering helps you design and write Tight structured programs following Warnier/Orr Methodology....
--Generates Code and Psuedo Code ....
--Easy to Learn --Easy to update....
--Full Featured Text Editor....

Software Support Intro Special

Pascal Version	C Version
~~$129.95~~	~~$149.95~~
$90.99	$104.95

Call Today 913-354-1618
701 Jackson B5 Topeka KS. 66603

CIRCLE 104 ON READER SERVICE CARD

C
how well
COMPUTER LANGUAGE
reaches and sells 55,000 professional software developers like yourself in the August

C Tools Issue

Call (415) 397-1881
by June 8
for space reservations

ISIS
CP/M MSDOS

COMPLETE SOURCE CODE INCLUDED!

ICXPDS: eXchanger now supports the 5 1/4"iPDS format. Manipulation of ISIS-II files using your computer system was never easier.
ICXMDS: Same as ICXPDS, but for MSD 8" systems.
IMXPDS: Reads/Writes 5" iPDS disks on PC's and AT's.
TELEDPLUS: Enhanced serial file transfer program for CP/M, ISIS, or MS-DOS.
ISE: Emulator gives the CP/M and MS-DOS user access to all the ISIS-II languages and utilities.
ACCELER 8/16: CP/M-80 emulator for MS-DOS. Enables PC's to run ISE. (no source code, V-20 ind.) $89 each $250 any 3 above
UDI: The 8086 ISIS Emulator runs all UDI applications. $300

ZAS Development Package: Z-8 and Z-8000 Assembler for CP/M, ISIS, and MS-DOS.

Request a catalog of our products!

Copyrights CP/M Digital Research, Inc.
ISIS-II and iPDS Intel Corp - MSDOS Microsoft

Western Wares 303-327-4898
Box C • Norwood, CO 81423

CIRCLE 105 ON READER SERVICE CARD

FORTRAN ADDENDA® v5.1

Comprehensive and easy to program I/O and graphics libraries. User I/O editing with context & type checking with Helps. Or enhance your graphics with TILEng. Tools for complete template systems and device independent and windowed graphics.

ASMUTIL2®	BUTILE®
• Full/FAST Graphics	• Natural I/O System
• Full PC, AT+ Keyboards	• Windows & Pull Downs
• Full Paths/Directories	• User Input Editing
• Music, Clocks, Strings	• I/O Error Recoveries
• RS-232s, CRTs, Printers	• No I/O Overflows

- ALL MDA/CGA/EGA/Hercules modes.
- IBM, Lahey, MS(3,4) and R-M Compilers. No fees.
- Extensive and easy to read manuals and samples.
- Argument checks, defaults and toggles to speed dev.

ASMULTIL2v5.1 $95; both $165. Updates to v5.1 $45.

IMPULSE ENGINEERING

Impulse Engineering, Ray Strong • (415) 788-4611
P.O. Box 3540 • San Francisco, CA 94119-3540

CIRCLE 101 ON READER SERVICE CARD

ADVERTISER INDEX

	PAGE NO.	CIRCLE NO.
2500 A.D. Software	83	27
Aker Corp.	95	40
Aldebaran Laboratories	25	16
Aspen Scientific	63	31
The Austin Code Works	126	68
BC Associates	123	67
BISS of Louisiana	133	75
Blaise Computing	13	7
Blaise Computing	5	3
Block Island Technologies	135	82
Borland International	1	1
Borland International	Cover II	—
The Boston Software Works Inc.	24	15
Breakpoint Computer Systems	133	78
Burton Systems Software	151	103
C Software Toolset	140	83
C Users' Group	151	102
C Ware/DeSmet	136	—
Cobalt Blue	133	76
Cosmos Inc.	26,27	17
Crackerjack Microsoftware Corp.	125	69
Creative Computer Software	143	91
Creative Programming	18	—
Database Applications	90	46
Datalight	12	6
Desktop AI	143	89
Ecosoft	113	62
Entelekon	52	51
ESP Software	133	77
FairCom	101	54
Farbware	135	81
Fillmore Systems Inc.	63	32
Flexus International	64	33
Genesis Data Systems	131	71
Gimpel Software	73	—
Gold Hill Computers	8,9	—
Greenleaf Software Inc.	44	24
Guidelines Software	17	10
Harvard Softworks	48	25
Hauppauge Computer Works Inc.	Cover III	107
Haventree Software Ltd.	111	60
Hedge Systems	20	11
Impulse Engineering	151	101
IMSI	91	47
Integral Quality Systems	23	13
Komputerwerk	143	90
Kurtzberg Computer Systems	100	53
Lahey	58	30
Lattice	108	59
Lifeboat Associates	57	29
Logic Path	32	20
Logitech Inc.	43	23
Lugaru Software Ltd.	90	45
Magus Inc.	55	28
Manx Software Systems	15	8
Mark Williams Co.	Cover IV	108
MDS Inc.	147	96
Meta Ware Inc.	117	64
Metagraphics	42	22
Microsoft	33	—
Microsoft	6,7	4
MicroWay	96	52
Mortice Kern Systems Inc.	147	97
N-TECH	151	104
Nanosoft	148	92
Norcom	93	49
Nostradamus Inc.	41	21
Oakland Group	78	36
Oasys	146	61
Opt-Tech Data Processing	151	99
P.M.I.	142	93
Polytron	10	5
Programmer's Connection	75,76,77	35
Programmer's Paradise	50,51	26
Programmer's Shop	121	66
Programmer's Shop	84,85	41,42
QCAD Systems	131	73
Quelo	151	100
Quilt Computing	140	84
Raima Corp.	86	43
Rational Systems	80	37
Realia	92	48
The Research Group	105	57
Resource Analysis International	131	74
Roundhill Computer Systems	28	18
Ryan-McFarland	119	65
Sapiens Software	135	79
SAS Institute	49	—
Scantel Systems Ltd.	135	80
Seidl Computer Engineering	134	94
Siltronix	131	72
SoftCraft	3	2
Softfocus	103	56
Software Connection Inc.	82	39
The Software Factory	141	85
The Software Family	145	95
Software Garden	94	50
Softway	143	88
Solution Systems	150	98
Solution Systems	106	58
Sophisticated Software	81	38
Structured Information Systems	151	106
Sydetech Systems Development Tech. Inc.	74	34
Texas Instruments	98	—
Turbo Power Software	138	87
Van Nostrand Reinhold	89	44
Vermont Creative Software	21	12
Visible Systems Corp.	114	63
Wendin Inc.	31	19
Western Wares	151	105
Whitesmiths	129	70
Wolf Pak Research	141	86
Workman & Associates	102	55
Yourdon Software Products	16	9

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

55,000

Professional Software Developers like yourself read this page— and the previous 150 pages. Reach and sell your product to them in the August issue of COMPUTER LANGUAGE

Includes a major wrap-up of

C Tools

Reservation deadline: June 8th

Contact your regional sales manager today:

East: (617) 232-5470
Midwest: (216) 493-8733
West: (415) 397-1881

HAUPPAUGE

Thrust your math into high gear!

HAUPPAUGE has a hunch you need faster math calculations. So we've launched our 287 FaST10—the highest speed math coprocessor available for PC/AT's and all AT compatibles.

Now you can speed up your math calculations without worrying about data safety. Our FaST10 has a self contained high speed clock that speeds up math coprocessor operations without jeopardizing accuracy.

The FaST10 accelerates programs including 1-2-3, Framework, AutoCAD and Symphony (to name just a few)—by as much as 250%!

Our FaST10 is compatible with: 6 or 8 MHz IBM PC/AT's, the Compaq DeskPRO 386 and all 286-based PC's like the Compaq DeskPRO 286 & Portable 286.

We have all the Coprocessors you need!

FaST modules, which operate at full speed, for the PC/AT, Compaq DeskPRO 386 and PC/AT compatibles:

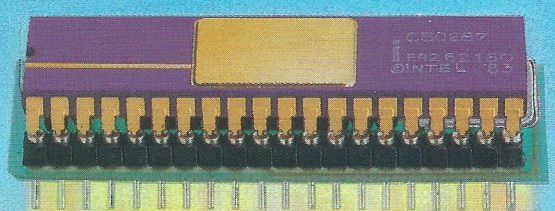
***287 FaST 10**—full speed 10MHz $469*
***287 FaST 8**—full speed 8 MHz $379*
***287 FaST 6**—full speed 6 MHz $249*

Math coprocessors for AT compatibles and accelerator cards:

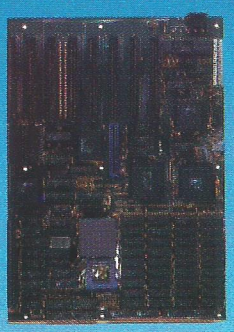
***287-10 Chip** $439*
***287-8 Chip** $349*
***287-6 Chip** $219*

Math coprocessors for PC/XT's & higher speed compatibles:

***87 Chip**—4.77 MHz ... $129*
***87-2 Chip**—8 MHz $195*



Save your PC from early retirement. Get Hauppauge's new 386 MotherBoard! *Hauppauge has a smart way to give your IBM PC or XT the speed and compatibilty of the 80386.*



It's our 386 MotherBoard. With a 32-bit, 16 MHz 80386 processor and 1 Mbyte of high speed memory, it scores high on desktop publishing, CAD graphics, engineering and business applications. 386 MotherBoard is a motherboard replacement that's 100% form fitting. So installation takes just 20 minutes.

To accept new generations of DOS, we gave it PC/AT BIOS, I/O compatibility and up to 16 Mbytes of high speed expansion RAM.

And our 386 MotherBoard has a lot more on the ball than simple accelerator cards. Because in addition to speed, it has a 16 bit expansion slot. Plus a 32 bit slot for high speed memory expansion. There's also a 387 math coprocessor socket for the ultimate in high speed math calculations, and a clock/calendar with battery backup for true PC/AT compatibility.

386 MotherBoard also comes with very intelligent prices:

***386 MotherBoard/PC** $1495*
***386 MotherBoard/XT** $1495*
***387 Math coprocessor** $695*

Hauppauge

Your math coprocessor specialists

Hauppauge Computer Works, Inc.
358 Veterans Memorial Highway,
Commack, N.Y. 11725
Phone: 516-360-3827

Trademarks: QuickBASIC: Microsoft, Calculus: AI Ware, HFT+: TechWare, MathTOOLBOX: Syntel, Labtech Notebook: Laboratory Technology Corp., PC and PC/AT: IBM Corp.

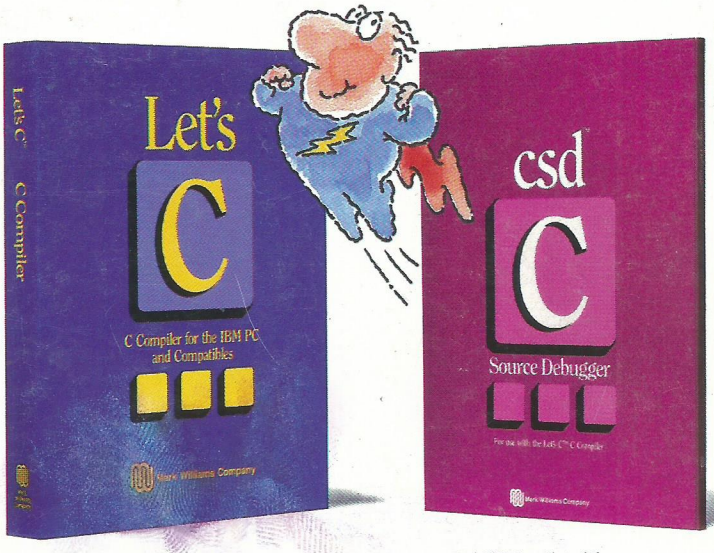
CIRCLE 107 ON READER SERVICE CARD

PRESENTING THE DIFFERENCE BETWEEN FAST COMPILING AND FAST PROGRAMMING.

For compiling speed, you can't do better than Let's C. But to really speed up programming you can't do without the powerful source level debugger, *csd*.

If you want the power, portability and flexibility of C, start with the complete compiler, Let's C. For utilities, editor, compiling speed and fast, dense code, Let's C has it all.

But to get your programs up and running you need more. Because even the fastest compiler can't outrun bugs. You need the revolutionary C Source Debugger, *csd*.



CUT DEVELOPMENT TIME IN HALF WITH *csd*

csd lets you bypass the time consuming frustrations of debugging–like long dumps and clunky assembler. With *csd*, you actually debug in C. You learn faster because you watch your program run in C. You finish faster because *csd* combines the speed of a compiler with the interactive advantages of an interpreter. The end result? Development time is sliced in half.

LIMITED TIME OFFER FREE *csd* WITH LET'S C!

LET'S C AND *csd* FEATURES

NEW VERSION 4.0!

Let's C:

- Now compiles twice as fast
- Integrated edit-compile cycle: editor automatically points to errors
- Includes both small and large memory model
- Integrated environment or command line interface
- 8087 sensing and support
- Documentation features new lexicon format
- MS-DOS object compatible
- New make utility
- Fast compact code plus register variables
- Full Kernighan & Ritchie C and extensions
- Full UNIX compatibility and complete libraries
- Many powerful utilities including make, assembler, archiver, cc one-step compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source included
- Supported by dozens of third party libraries
- For the IBM-PC and Compatibles
- Not copy protected

Sieve Benchmark

(Compile time in seconds)
Let's C: **2.8** (On 512K 6Mhz IBM-AT)
Turbo C: **3.89** (As advertised)

csd:

- Large and small memory model
- Debug in C source code, not assembler
- Monitor variables while tracing program
- Does not change program speed or size
- Provides separate source, evaluation, program and history windows
- On-line help screens
- Can interactively evaluate any C expression
- Can execute any C function in your program
- Trace back function
- Ability to set trace points
- Not copy protected

REVIEWERS ARE RAVING ABOUT LET'S C AND *csd*.

"Let's C is an inexpensive, high-quality programming package... with all the tools you will need to create applications."
–William G. Wong, *BYTE*, August 1986.

"The performance and documentation of the $75 Let's C compiler rival those of C compilers for the PC currently being sold for $500...highly recommended..."
–Marty Franz, *PC TECH JOURNAL*, August 1986.

"csd *is close to the ideal debugging environment...a definite aid to learning C and an indispensable tool for program development."*
–William G. Wong, *BYTE*, August 1986.

"This is a powerful and sophisticated debugger built on a well-designed, 'serious' compiler."
–Jonathon Sachs, *Micro/Systems Journal*, April, 1986

START TO FINISH, THERE'S NO BETTER ENVIRONMENT.

Get started with the right C compiler and you'll have everything you need for development–including source level debugging. On top of it all, Let's C and *csd* are today's best values in professional C programming tools. And most reliable: Mark Williams C compilers have been sold with DEC, Intel and Wang computers since 1981.

60 DAY MONEY BACK GUARANTEE

Mark Williams gives you a full 60 days to find out just how good Let's C and *csd* really are–or your money back.

So if you want more than a fast compiler–if you want your programs up and running fast, ask for Let's C and *csd*. You'll find them at your software dealer's, in the software department of your favorite bookstore, through the Express Program at over 5500 Radio Shacks or you can order now by calling 1-800-MWC-1700.*

*In Illinois call, 1-312-472-6659.

Mark Williams Company

1430 West Wrightwood, Chicago, Illinois 60614

© 1987 Mark Williams Company
Let's C is a registered trademark of the Mark Williams Company.
UNIX is a trademark of Bell Labs.

MARK WILLIAMS LET'S C AND *csd*. ONLY $75 EACH.

CIRCLE 108 ON READER SERVICE CARD